uc3m | Universidad Carlos III de Madrid

Ingeniería Electrónica Industrial y Automática
2017-2018

Trabajo Fin de Grado

"UAV Localization in Grid-Style Minesweeper Areas"

Gonzalo Cabanas Yuste
Tutor/es
Abdulla Hussein Abdulrahman Al-Kaff

“El verdadero progreso es el que pone la tecnología al alcance de todos”
- Henry Ford

AGRADECIMIENTOS

En primer lugar, quiero agradecerle la elaboración de este Trabajo de Fin de Grado a mi familia. Gracias por enseñarme que con esfuerzo se consigue cualquier cosa que me proponga; que se puede anteponer los viajes, fiestas y amigos a los exámenes y estudios porque solo se vive una vez, siempre que luego se cumplan con ellos. No puedo pedir una familia mejor, ni agradeceros lo suficiente todo lo que me habéis dado, pero espero que este trabajo os enorgullezca.

En segundo lugar, quiero agradecer a mis amigos la dedicación que han tenido conmigo. P.D.P, os llevo siempre a mi lado.

En última instancia, quiero agradecer la elaboración de este Trabajo a mi abuelo. Abuelo, siempre presumías de mi a terceros, y mostrabas el orgullo que era tenerme como nieto. Corrígeme al decirte que el orgullo ha sido mío por tenerte, y si bien no te ha dado tiempo a verme acabar la carrera, que sepas que has estado siempre presente en mí día a día. Descansa en paz, te lo mereces…

RESUMEN

En el presente trabajo se pretende desarrollar un algoritmo para la detección de celdas en un terreno con minas y la generación de rutas seguras para la desactivación de las minas localizadas en el terreno, mediante el uso de un UAV que genera imágenes aéreas del terreno.

Para ello se partirá de los principios de odometría visual, para filtrar la imagen entrante de todo ruido posible. Posteriormente, se enviarán las celdas detectadas mediante el Sistema Operativo Robótico “ROS” para poder comunicarse con otros componentes del proyecto y localizar las celdas que contengan minas.

Finalmente, se generan rutas solución por cada celda que contenga minas, con el fin de crear una ruta segura para la desactivación correcta de las minas.

ABSTRACT

In this Project, it is proposed to develop an algorithm for the recognition of grids in a field with landmines, and also the generation of safety-routes for the desactivation of the landmines located in the field, using a UAV for taking aereal pictures of the terrain.

The algorithm is based on visual algorithms to filter the inputs images from the digital noise. Next to this, it will be send it the detected grids with the use of an Robotic Operating System (ROS), which makes possible the communication with the other devices in the proyect. ROS will also make possible to know which grids have landmines on it.

Finally, the algorithm will create safety-routes for any grid which has landmines on it, with the purpose of desactivate those landmines.

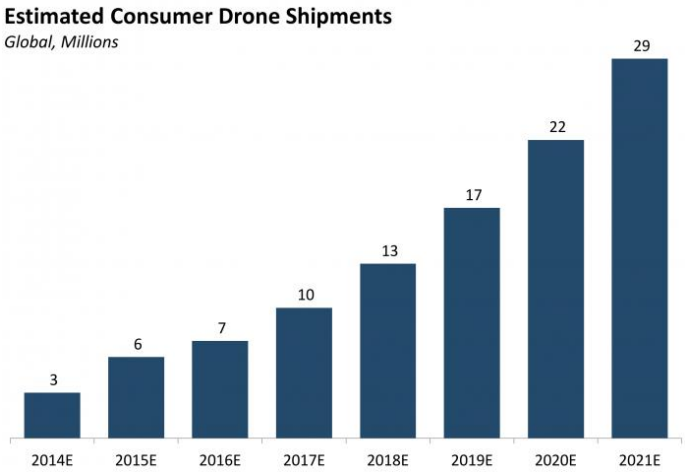
INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES

1. INTRODUCCION

Los Vehículos Aéreos no Tripulados (UAV, siglas inglesas de Unmanned Aerial Vehicles), han ganado un hueco en el mercado como uno de los sectores más fiables para la inversión y desarrollo. Tal es su éxito que cada vez es más frecuente ver estos vehículos autónomos sectores inconcebibles hasta el momento.

Podemos encontrar estos UAV's, conocidos coloquialmente como drones, implementados en sectores como agricultura, salvamento, telecomunicaciones, transporte, entretenimiento… Tales son las aplicaciones que pueden involucrar el uso de drones, que cada año aumenta exponencialmente la manufacturación y producción de estos vehículos.



Andrew Meola (2016). Ilustración 1. Estimación de Envíos de UAV's. Recuperado de https://www.businessinsider.com.

Si bien se está produciendo un aumento en la producción de UAV's, causada mayoritariamente por la expansión de los vehículos en los sectores mencionados anteriormente, también se ha notado una mayor producción de drones en el sector donde se originaron los primeros prototipos de estos vehículos aéreos no tripulados: el sector de defensa y seguridad.

La mayoría de avances tecnológicos que tenemos en el sector civil provienen en gran medida de investigaciones y tecnologías derivadas del sector de la defensa. Esto se debe a que cada vez se busca implicar menos al factor humano en conflictos bélicos, hasta tal punto que se ha desarrollado tecnologías pioneras que facilitan la seguridad activa del soldado. El desarrollo de la tecnología de ámbito militar consigue mejorar exponencialmente cualquier tipo de operación militar, ya sea logística, reconocimiento, armamentística, humanitaria… Estas mejoras implican también un mayor cuidado a la hora de realizar cualquier tarea que implique el despliegue de efectivos humanos, debido a que cualquier bando dispone de las mismas, o parecidas, condiciones tecnológicas que afectan considerablemente a la seguridad del personal.

Tanto las guerras actuales como las de los últimos años han dejado una gran cantidad de residuos explosivos sin detonar, que cada año provocan la muerte de miles de personal.

Según la Campaña Internacional para la Prohibición de las Minas Antipersona (ICBL, pos sus siglas en inglés) cada día pierden la vida 10 personas a causa de la explosión de minas terrestres y resto de tecnología militar utilizado en estos conflictos bélicos. Este peligroso armamento provoca accidentes tanto en el sector civil como en el militar, por lo que supone una gran necesidad gestionar y aumentar la seguridad activa de las personas cercanas a las minas.

El proposito de este Trabajo de Fin de Grado es acercar una solución, utilizando tecnologías militares actuales, a la desactivación de minas personales en terrenos conflictivos, sin tener que desplegar ningún efectivo humano sobre el terreno. Con esta idea, se garantiza la seguridad activa del personal, tanto civil como militar, a la hora de desactivar dichas minas.

Este proyecto forma parte de un conjunto de estos para una competición que se realiza en Septiembre de 2018. La competición consiste en la utilización de vehículos autónomos para la desactivación de minas personales en un terreno clasificado en celdas. Para esta competición, hacemos uso de tres componentes: un UAV, un Rover o vehículo terrestre no tripulado, y un terminal en tierra (PC). En la siguiente figura se muestra un esquema del despliegue de los componentes, y su funcionamiento general:

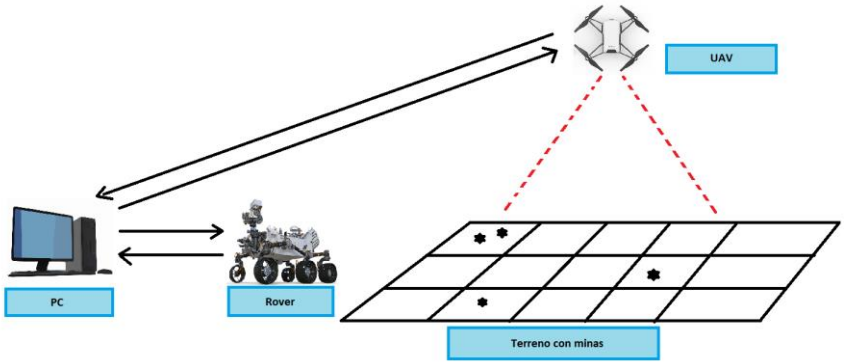


Ilustración 2. Despliegue general del proyecto

El proyecto general destinado a la competición consistiría en los siguientes apartados:

- El uso de un dron para la clasificación y detección de un terreno pintado con celdas, detección de minas personales en cada celda y generación de rutas seguras para la desactivación de dichas minas
- El uso de un Rover para la desactivación de las minas personales a través de las rutas seguras generadas por el dron
- Un terminal desde el que se pueda asistir la detección y desactivación de las minas personales, recibiendo la información llegada del dron y enviando las rutas seguras al Rover.

Dentro de las funcionalidades a desempeñar por el dron, este Trabajo de Fin de Grado se centra en la clasificación y detección de las celdas de cualquier terreno irregular, además de la generación de rutas seguras para la desactivación de las minas, por parte del Rover.

El Trabajo está estructurado en Capítulos, que se organizan de la siguiente manera: en el Capítulo 1, en el que nos encontramos, hacemos una pequeña introducción al proyecto; En el Capítulo 2, hablamos sobre la historia del arte relacionada con el origen y tipos de drones, proyectos similares al TFG o artículos útiles para la redacción del trabajo; El Capítulo 3 trata de la explicación general de la solución propuesta; El Capítulo 4 muestra el diagrama de flujo general del algoritmo y resultados experimentales; el Capítulo 5 resume líneas futuras del trabajo y conclusiones; El Capítulo 6, perteneciente a los Anexos, enseña el código utilizado para la el TFG, además de esquemas de funcionalidades internas del algoritmo; por último, el Capítulo 7 es una bibliografía con páginas web visitadas.

Espero que la lectura de este Trabajo de Fin de Grado resulte sencilla, y agradecer al lector por el interés mostrado en el proyecto.

2. ESTADO DEL ARTE

Con el incremento en la producción y desarrollo de nuevos UAV's, cada vez son más los sectores que hacen uso de estos vehículos aéreos no tripulados para desarrollar proyectos de una alta innovación y complejidad.

La adaptación de estos drones para uso doméstico propicia la investigación y utilización de este tipo de tecnología para mejorar el día a día de la sociedad.

Este uso de vehículos aéreos no tripulados, combinado con tecnologías de visión por computación, abre las puertas a una innovación de los sectores todavía mayor, pues al utilizar técnicas de procesamiento de imágenes, podemos expandir los usos y aplicaciones que pueden tener los UAV's

2.1 Visión por computación y uso de UAV's destinado al sector de agricultura

Investigadores de universidades españolas (IAS de Córdoba, ICA de Madrid y ETSEA de Lleida) realizaron un estudio con el uso de un UAV, que consistía en la detección de infestaciones de grama (*Cynodon dactylon*), en terrenos con viñedos. Mediante el uso de un dron convencional, al que se le une una cámara de fotos personal, se hizo un barrido para mapear desde el aire un terreno dedicado a la viña. El objetivo de la investigación era crear un método que permitiese localizar la aparición de esta mala hierba, para reducir considerablemente los daños por perdidas de cosecha. Los resultados fueron muy prometedores, tales que se llegó a conseguir una fiabilidad total del 83,3% en la detección de grama, como se muestra en la siguiente figura:

Cobertura de grama	Exactitud en la clasificación
Baja < 5 %	100%
Media 5-30 %	71,4%
Alta > 30%	80%
Fiabilidad global	83,3%

Ana I. de Castro (2018). Ilustración 3. Resultados sobre investigación de campo. Recuperado de http://www.interempresas.net

Yamaha General Motors lleva desde la década de los 80 desarrollando aplicaciones agrícolas en las que involucra el uso de drones. Un ejemplo es el desarrollo de un dron capaz de detectar las zonas de cultivo de arroz que requieren de una fumigación, y las marca en un mapa virtual. Es el propio agricultor quien, después de marcar los puntos, envía de manera autónoma al UAV a fumigar los puntos seleccionados. Con el uso de esta tecnología se eliminan los costes derivados para la contratación de personal de fumigación, a la vez que se detecta de una manera más eficaz aquellas zonas que requieren cuidados urgentes.

Una Start-Up localizada en la Universidad Politécnica de Varsovia, Polonia, lleva investigando desde 2011 el uso de microdrones, combinados con técnicas de visión por computación, para la polinización de las plantas, cultivos, arboles salvajes… Este proyecto surge ante el peligro que supone la desaparición de las abejas en el planeta. Ya decía Einstein: "Si la tierra se quedara sin abejas, el ser humano desaparecería en 4 años". Un test realizado en 2016 resulto para el proyecto un rotundo éxito, tras imitar con eficacia la polinización natural.

Opublikowano (2016). Ilustración 4. Micro dron realizando pruebas de polinización. Recuperado de https://www.pw.edu.pl

2.2 Visión por computación y uso de UAV's destinado al sector de salvamento

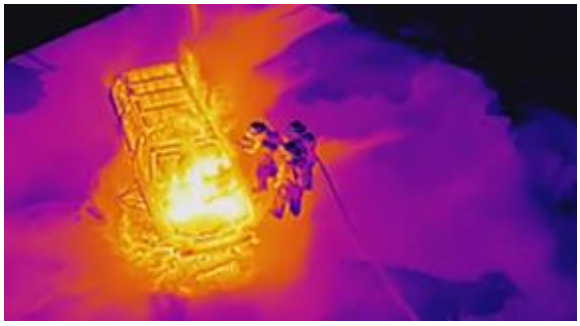
La Universidad de Huelva (UHU) patentó el pasado Agosto un sistema de salvamento que involucra la utilización de drones en la costa. El proyecto consiste en la asistencia al salvamiento y rescate mediante el guiado por láser y GPS de un dron que lleva como carga un chaleco salvavidas. Es el socorrista quien, en caso de accidente, guía al dron

con un láser hasta el punto donde se encuentra la víctima. El dron, a parte de los chalecos salvavidas, está dotado de un altavoz que permite la comunicación del socorrista desde el UAV hasta la persona a rescatar. Al dotar de unos prismáticos al dron, este identifica el lugar de riesgo para determinar el recorrido a realizar.

Flyability, una empresa dedicada a la fabricación de UAV's industriales, desarrolló un dron que facilitaba la búsqueda y rescate de personas en zonas peligrosas de difícil acceso, tales como grietas en placas de hielo, terrenos con movimientos de tierra o cenagales, entre otros…El dron está dotado de una capsula que amortigua los golpes y protege al UAV de los componentes más delicados, como las hélices. El dron está dotado de una cámara de video que transmite a tiempo real lo que ve, para que el usuario del UAV pueda encontrar a la víctima.

Lausanne (2014). Ilustración 5. Dron de salvamento con su carcasa protectora. Recuperado de https://www.startup.ch/flyability

FlightTech Systems es otro ejemplo de compañía que utiliza la combinación de drones con tecnologías de procesamiento de imágenes. Ha desarrollado un dron que, equipado con cámaras de vigilancia y cámaras térmicas, ayuda a la detección de incendios forestales y gracias a su vigilancia autónoma permite un control de las zonas más secas de la Península Ibérica. Utilizando un GPS y la cámara de vigilancia, mapean una zona determinada por personal cualificado. Las cámaras térmicas toman fotografías en las que se puede analizar la presencia de fuego o no. En caso de detectar fuego, se manda una alarma a la estación situada en tierra con la localización exacta del foco del incendio.



Zenmuse (2017). Ilustración 6. Imagen de una cámara térmica en modo reconocimiento. Recuperado de https://www.dji.com

2.3 Visión por computación y uso de UAV's destinado al sector de Seguridad Civil

Safran electronics tiene un sistema de seguridad, denominado Patroller, que se encarga de la implementación de una flota de UAV capaces de llevar funciones de reconocimiento de terreno enfocados en la seguridad de infraestructuras. Equipados con cámaras de últimas tecnologías, son capaces de detectar rostros para identificar personal autorizado e intrusos en el terreno, para llevar a cabo las medidas de seguridad pautadas.

La empresa Microdrones es otra de las que hace uso de la visión por computación, combinada con el uso de drones, para implementar sistemas que ayudan en la seguridad de los civiles. Ha desarrollado un sistema de seguridad capaz de reconocer rostros de personas afines a grupos terroristas, para detenerlas antes de que se produzca algún intento de golpe terrorista. El UAV sobrevuela de manera autónoma una localización, y en el momento en el que detecta un objetivo conflictivo, mantiene un rumbo fijo, tomando como referencia la propia persona detectada. Una vez verificada la amenaza, se llama a los servicios de seguridad pertinentes, tales como policía o ejército, para detener al individuo.

Microdrones (2017). Ilustración 7. Micro dron para Seguridad. Recuperado de https://www.microdrones.com

Uno de los sistemas de seguridad más atractivos que combina el uso de drones y visión por computación, es el sistema aportado por la empresa Airobotics. Con una flota de drones alrededor de un perímetro con estructuras a proteger, estos son capaces no solo de detecta intrusos no autorizados en el perímetro, sino que además pueden detectar fallos estructurales o fugas peligrosas en los edificios, por lo que incrementa la seguridad activa y pasiva de los civiles. Tienen además un centro de control autónomo, desde donde se cargan y gestionan automáticamente todos los UAV's, por lo que la interacción hombre-dron se reduce a la alerta que emite el dron cuando detecta una amenaza, ya sea en la infraestructura del edificio como la detección del intruso.

2.4 Visión por computación y uso de UAV's destinado al sector de logística

El gobierno británico ha llegado a un acuerdo con la gigante Amazon para el empleo de UAV's, combinados con la tecnología de visión artificial, para realizar pruebas de entregas en zonas suburbanas o rurales. Bajo el nombre de Amazon PrimeAir, los

drones están dotados de cámaras para poder detectar y evitar aviones, edificios y personas de forma automática. Una vez llega al destino, deja el paquete cuidadosamente, reduciendo a intervalos de minutos-horas un reparto que de forma convencional podría tardar varios días en realizarse. Tal es el afán de reducción de tiempos de entrega que se ha desarrollado un UAV específico para dichas tareas logísticas. Hibrido entre un avión y un helicóptero, su diseño permite realizar vuelos de distancias largas a un coste muy bajo.

Amazon Prime Air Company (2018). Ilustración 8. UAV de Amazon Prime Air. Recuperado de https://www.amazon.com

Un proyecto de una Start-up está realizando un estudio parecido al de Amazon para utilizar UAV's en la gestión de ayuda humanitaria. El proyecto consiste en crear una red de vehículos aéreos no tripulados para entregar suministros médicos, alimentos y otros materiales a las personas en zonas rurales de distintos países del continente africano. Gracias a este proyecto, se consigue el acceso de necesidades básicas en los lugares donde, por causas del entorno, se dificulta mucho el acceso para vehículos convencionales, ya sea el corte de una carretera por lluvias o la localización del poblado en el interior de una selva.

En Enero de 2018, Boeing presenta un prototipo de UAV de carga capaz de soportar hasta 226 kg. Este UAV supone una revolución en el sector de la logística, pues manejar cargas de pesos tan elevados con un dron agiliza los tiempos de entrega de mercancía y los movimientos internos de carga en una fábrica. Al tratarse de un UAV de conducción autónoma, está dotada de las últimas tecnologías, entre otras aquellas dedicadas al procesamiento en tiempo real de imágenes, para detectar y esquivar obstáculos durante su vuelo.

2.5 Visión por computación y uso de UAV's destinado al sector militar

Tomando como referencia los UAV's en el área militar, es muy común que el primero que se nos venga a la cabeza sea el UAV Reaper, vehículo aéreo no tripulado utilizado por las tropas de Estados Unidos. Se concibió como el primer dron de ataque diseñado para misiones de vigilancia de larga duración y gran altitud, de manera completamente autónoma, gracias a su Sistema de adquisición de blancos AN/DAS MTS-B, que está basado en técnicas de procesamiento de imagen.

Ministerio de Defensa (2017). Ilustración 9. UAV Reaper, en vuelo. Recuperado de https://www.defensa.com

El Gobierno chino ha estado investigando durante los últimos años sobre la utilización de drones de combate para la incorporación de estos en el ejército. Contando con las ventajas que ofrece la visión artificial, se dio a conocer a finales del año pasado el desarrollo del UAV Sharp Sword ("Liajin" en su idioma natal, el chino mandarín). Estos drones son capaces de llevar tareas de reconocimiento de terreno, personal enemigo y objetivos críticos, entre otros. Con una capacidad de carga de 2 toneladas de material explosivo, también pueden utilizarse para tareas de combate directo.

Cheyenne MacDonald (2017). Ilustración 10. UAV Sharp Sword, en vuelo. Recuperado de https://www.dailymail.co.uk

En Julio de 2017 se presenta en el Salón Aeronáutico MAKS, en Zhukovsky (Rusia), el UAV Orion-E de Media altura y larga duración. Con una autonomía máxima de tiempo de vuelo de 24 horas, este UAV se concibe para misiones que tengan que ver con la vigilancia, inteligencia y reconocimiento de objetivos. Dicha detección de blancos se consigue gracias a una carga electro-óptica que es capaz de tener un rango de visión directa de 250 kilómetros.

Ernesto Santana (2018). Ilustración 11. UAV Orion, en vuelo. Recuperado de http://www.xdrones.es

Un último ejemplo de UAV aplicado al ámbito militar y que combina técnicas de visión por computación se está desarrollando en tierras españolas. En el Instituto Nacional de Técnica Aeroespacial (INTA), se lleva investigando la creación de un dron capaz de adaptarse a cualquier tipo de misión pensada para un UAV según el material que se le instale en él. Así, nace la idea del UAV Milano. Está concebido para ser "el chico para todo": labores de vigilancia de objetivos, control de incendios, labores de rescate, control de cosechas, combate directo… tendrá una autonomía de 20 horas, y podrá llevar cerca de 1000 kilos de carga útil (armamento, radares, cámaras, sistemas anti-drones). Actualmente se encuentra en fase de desarrollo y pruebas de vuelo.

Iñaki de la Torre (2013). Ilustración 12. Prototipo del UAV Milano. Recuperado de https://www.quo.es

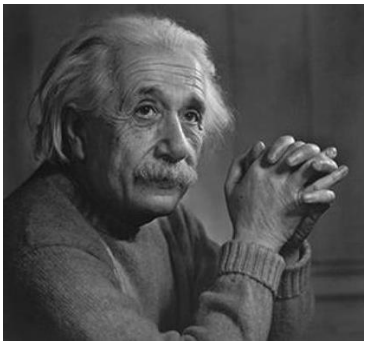
3. SOLUCION PROPUESTA Y EXPLICACION DEL ALGORITMO

3.1 Conversión imagen a espacio HSV

Nuestro flujo de programa comienza con la entrada de una imagen digital que representa el terreno con celdas a clasificar.

Una imagen digital consiste en una combinación de píxeles, agrupados en una matriz bidimensional, que toman un valor determinado. Según la representación de color para la imagen, diferenciaremos 2 tipos: imágenes en blanco y negro e imágenes a color.

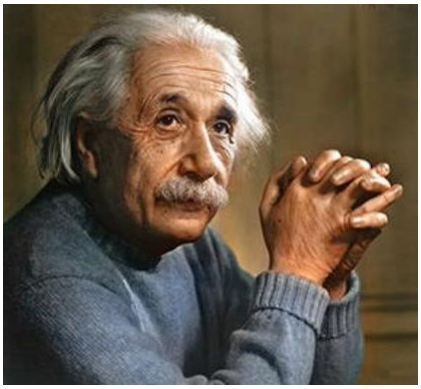
La forma más sencilla de representar la realidad mediante imágenes, es haciendo dicha representación digital con imágenes en blanco y negro. Esto se debe a que una imagen blanco-negro es una combinación de pixeles en forma de array que van a tomar un valor de gris en función de la tonalidad del color a representar. Esto quiere decir que si el color real tiende a tener tonos muy oscuros, el píxel tiende a escoger un valor próximo al color negro ('0' lógico), mientras que si el color tiende a tonos más claros, el pixel que le representara se aproxima al color blanco ('1' lógico).



(2015). Ilustración 13. Imagen digital en blanco y negro. Recuperado de http://infoblancosobrenegro.com/

Si bien es cierto que una imagen en escala de grises (blanco y negro) tiene un coste computacional bajo, la representación con la realidad es pésima, pues solo se tiene en cuenta la tonalidad (oscura o clara), dejando a un lado los colores. Haciendo uso de mayor coste computacional, podemos digitalizar la realidad mediante imágenes a color.

Una imagen a color es una combinación de tres matices pixeles que se combinan entre sí para poder representar, matemáticamente, el color deseado. Dependiendo del formato de color, dichas matrices harán referencia a diferentes características que definen a un color, y es la variación del valor del pixel, junto con la combinación de las otras matrices, que dan cabida a la representación de toda la gama de colores del espectro luminoso visible para el ser humano.

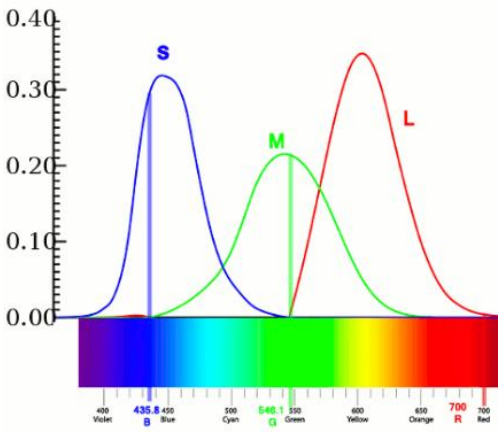


(2015)Ilustración 14. Imagen digital en color. Recuperado de https://ar.pinterest.com

En nuestro algoritmo, trataremos con imágenes de formato RGB, que serán convertidas al espacio HSV.

El formato de representación RGB (Red Green Blue) parte del mecanismo de funcionamiento del ojo humano. En la retina del ojo humano hay una serie de neuronas modificadas, que se encargan de la percepción del color del mundo físico. Estas neuronas se dividen en dos tipos: bastones y conos. Mientras que los bastones se encargan de la percepción de la luminosidad del ambiente, son los conos los que se encargan de percibir los colores.

Encontramos tres tipos de conos para captar la información del color observado: de tipo L, tipo M y tipo S. Dichos receptores se encargan de captar la luz oscilante en rangos de longitud de onda distintos, obteniendo como información el color que se está observando. Así, el cono tipo L abarca el rango de colores con longitud de onda cercana a la del color rojo (colores rojizos); el de tipo M a colores verdosos y por último, el de tipo S, a colores cercanos al tono azul. Son precisamente a estos colores: Rojo, Verde y Azul, a los que tenemos mayor sensibilidad para detectarlos, por lo que los veremos con mayor intensidad.



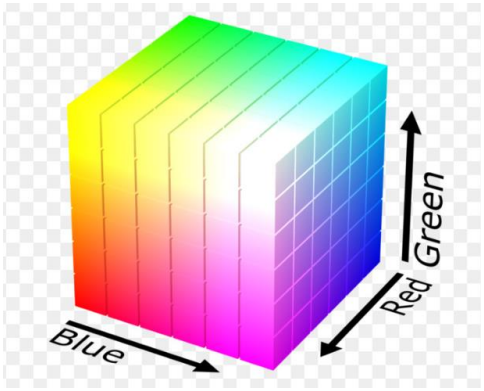
(2013) Ilustración 15. Longitud de onda frente a sensibilidad normalizada de los conos ópticos. Recuperado de http://www.pensamientoscomputables.com

Fue finalmente en la década de 1920 cuando la CIE (Comisión Internacional de Iluminación) escogió estos tres colores (RGB), como los representantes de los rangos de color captados por un cono, debido a que la longitud de onda de cada uno de estos tres colores solo puede ser captado por su cono asociado.

Partiendo del mecanismo de funcionamiento del ojo humano, y una vez establecidos los principales colores para la representación de todo el rango de colores del espectro visible, surgió el formato de representación RGB para imágenes a color.

El formato RGB, de las siglas Red-Green-Blue, combina las tres matrices de pixeles para cada uno de los tres colores del formato. Como los pixeles pueden alterar su valor de color (de 0 a 255), produciendo una mayor o menor tonalidad dependiendo del valor, se puede obtener todo el rango de colores del espectro de luz visible combinando distintos valores de pixel entre sí.

Es un método rápido y eficaz para digitalizar el color de la realidad que se quiere representar, y es el formato de representación utilizado por excelencia en monitores, televisiones, pantallas… satisfaciendo la representación del color para el ojo humano. Pero es en el área de la visión artificial cuando dicha forma de representar el color nos puede limitar a la hora de obtener resultados fiables, o poder manipular mejor los datos extraídos que dependen directamente del color de la imagen. Concretamente, el formato RGB tiende a dar problemas cuando se requiere extraer información de colores cercanos al blanco, pues al ser el blanco es la saturación total de los píxeles de los tres canales, no sé pueden distinguir correctamente distintas tonalidades de este color.



(2012). Ilustración 16. Cubo RGB. Recuperado de https://www.pinterest.es

En nuestro caso particular, el formato RGB no nos sirve de nada, pues como necesitamos extraer como información las celdas del terreno, que están pintadas de color blanco, tenemos que usar otro formato de color para representar la imagen y conseguir facilitar la detección de dichas celdas.

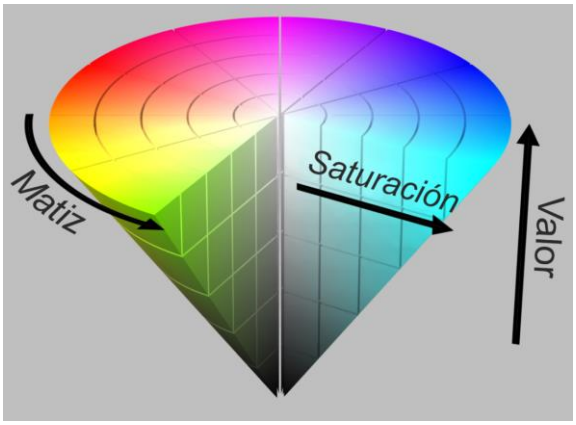
Como evolución al modelo de color RGB se crea en 1978, a manos de Alvy Ray Smith, una nueva forma de representar el espectro luminoso visible: el Modelo de color HSV. Este modelo surge como una transformación no lineal del modelo RGB.

El modelo HSV enfoca la forma de representar un color de una manera completamente distinta al Modelo RGB. Si bien el Modelo RGB combinaba distintos valores de pixeles de los colores Rojo-Verde-Azul para representar un color, el modelo HSV va a tomar como características para representar un color: El tinte o matiz; la saturación y el brillo. Se llama Modelo HSV por las siglas Hue (tinte/matiz), Saturation (saturación) y Value (brillo/valor).

Según el Modelo de color HSV, el tinte hace referencia al color a representar en sí. Esto quiere decir que ya no combinaremos distintos colores para obtener otro, si no que tendremos un valor seleccionado de tinte para cada color del espectro luminoso visible (o prácticamente su totalidad).

La saturación de un color se puede definir como su intensidad. Colores con saturación baja significan colores de tonalidades grisáceas, mientras que colores con saturación alta representan colores más “puros”.

Por último, el valor del brillo se puede traducir como la representación del blanco-negro. Un color con un brillo al 100% puede representar colores muy claros y próximos al color blanco.



MaulucioniI. Ilustración 17. Cono HSV. Recuperado de https://es.wikipedia.org

Mientras que el Modelo de color RGB sigue una representación gráfica en función de las coordenadas euclideas, el Modelo de color HSV se rige por una representación gráfica con coordenadas cilíndricas. Es un modelo que si bien a simple vista puede parecer que no termine de representar correctamente la realidad, facilita mucho al algoritmo para encontrar los tonos blancos, pues quedan representados matemáticamente mejor que el Modelo RGB (el Modelo RGB se utiliza para las pantallas, y es por eso que nos resulta más fiable).

Así, por cada imagen que entre en el código, que siempre tendrá el formato RGB, se le realiza una conversión al espacio HSV, antes de realizar cualquier operación en la propia imagen.

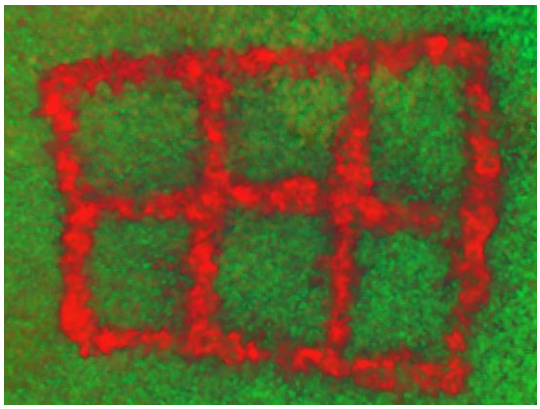


Ilustración 18. Celdas en el espacio de color HSV.

Aunque a simple vista pueda parecer que el resultado tras aplicar la conversión al espacio HSV sea peor, para al algoritmo resulta más sencillo representar la imagen a color de esta forma, para luego separar correctamente los colores que se necesiten.

3.2 Preprocesamiento de la imagen. Filtro de la mediana.

Una vez convertida la imagen al Modelo de color HSV, nos encontramos con el siguiente problema: las celdas están dibujadas en terreno exterior, por lo que tiene contacto directo con la luz solar. Si bien es verdad que al ser una imagen estática la luz

solar no produce muchos problemas a la hora de detectar correctamente el entorno, tales como sombras en zonas aleatorias, reducción de la calidad de imagen…los rayos de luz provocan reflejos en el terreno que hacen saturar los píxeles al color blanco, color que estamos buscando separar del resto de colores, para detectar las celdas. Es por este problema que aparece mucho ruido al intentar separar el color blanco, como se aprecia en la imagen:

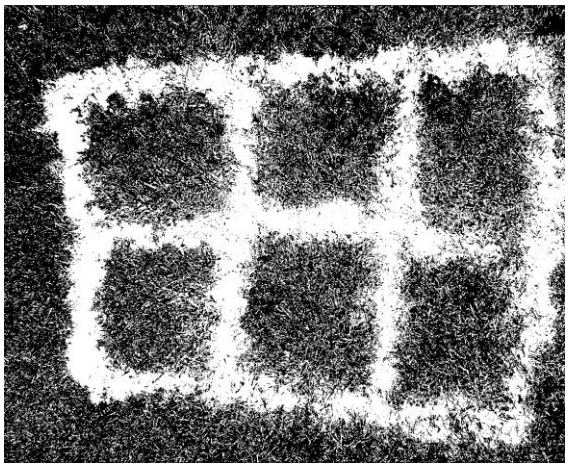


Ilustración 19. Imagen binaria sin aplicar pre procesado.

El ruido en una imagen es una serie de píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos. El ruido en una imagen puede venir de factores externos, tales como el exceso de luz solar, condiciones atmosféricas, variaciones bruscas de iluminación… además de factores internos, como la alimentación de la cámara, la propia lente o la digitalización de la imagen, entre otros.

Podemos encontrar varios modelos de ruido, que aun siendo el mismo concepto, se distribuyen en función de densidades de probabilidad distintas. Destacamos:

- Ruido Gaussiano: Se produce involuntariamente por los circuitos electrónicos propios de la cámara. Siempre tendremos este tipo de ruido en cualquier imagen.

Lidia Belén Vega (2016). Ilustración 20. Ejemplo de ruido Gaussiano. Recuperado de https://docplayer.es

- Ruido Impulsional: Este ruido está más relacionado con la cuantificación realizada en el proceso de digitalización de imagen. Se traduce con pixeles aleatorios aislados que se saturan en color blanco (1 lógico) o negro (0 lógico).

Lidia Belén Vega (2016). Ilustración 21. Ejemplo de ruido impulsional. Recuperado de https://docplayer.es

En la imagen sin filtros (Ilustración 19), se puede observar cómo se ha llegado a reconocer correctamente las celdas del terreno. Sin embargo, los reflejos provocados por la luz solar han afectado a la detección de estas celdas, apareciendo pequeños objetos considerados como objetos en la imagen, aunque sea ruido. Un objeto es aquello que el algoritmo toma como dato para procesar (todo lo que aparece en blanco, en la imagen binaria). Lo demás es fondo, datos que no nos sirven y que son excluidos.

Para solucionar este problema, tenemos que realizar algún tipo de filtrado digital que permita eliminar el ruido de la imagen y a la vez asegurar que las celdas no dejan de ser detectadas.

Los filtros digitales son operaciones matemáticas que tienen como objetivo encargarse de suavizar la imagen, eliminar ruido, realzar las características propias de la imagen y detectar sus bordes.

Hay dos tipos de filtros digitales, en función del dominio por el que sea realice dicho filtrado: Filtros pertenecientes al dominio del espacio y filtros pertenecientes al dominio de la frecuencia. Los filtros pertenecientes al dominio del espacio trabajan directamente sobre los pixeles de la imagen, mientras que los pertenecientes al dominio de la frecuencia realizan operaciones basadas en la transformada de Fourier de la imagen. Nos centramos en el dominio del espacio, debido a que no se han producido ningún tipo de ruido que requiera hacer uso de filtros del dominio de la frecuencia.

Dentro de este dominio, podemos encontrarnos con dos tipos de filtros: lineales y no lineales. Es de vital importancia reconocer bien el ruido que queremos eliminar para poder escoger entre un tipo de filtro u otro. Volvamos a la Ilustración 19.

Como bien se ha explicado antes, el problema que tenemos en la imagen es que están saturándose pixeles debido a la luz solar directa, que provoca reflejos aleatorios en el césped. El tipo de ruido que más se asemeja a la situación es el ruido impulsional, y será este tipo de ruido el que queramos eliminar.

Los filtros lineales hacen uso de una matriz máscara, en la que viene definido el modelo matemático a aplicar en los pixeles. Dicha mascara se combina con la imagen original

para obtener una imagen resultante. El tipo de filtrado está determinado por el contenido de dicha máscara.

Los filtros lineales son eficaces para eliminar el ruido gaussiano, además de tener un coste computacional bajo. El inconveniente del uso de estos filtros es que no funcionan en el caso de que aparezcan cambios locales aislados, que es precisamente el que produce el ruido impulsional, además de emborronar mucho la imagen, difuminando los bordes. En este caso, ya no nos serviría de nada para el algoritmo.

Los filtros no lineales no hacen uso de operaciones para eliminar el ruido. En este tipo de filtros, prevalece el uso de la estadística del orden. Esto quiere decir que en estos filtros ordenaremos los valores en la vecindad de cada pixel, para obtener un valor nuevo del pixel a partir de la lista ordenada.

En este tipo de filtros, destacamos tres:

- Filtro del mínimo: que selecciona el valor más pequeño del conjunto de pixeles para que el píxel seleccionado tome ese nuevo valor.

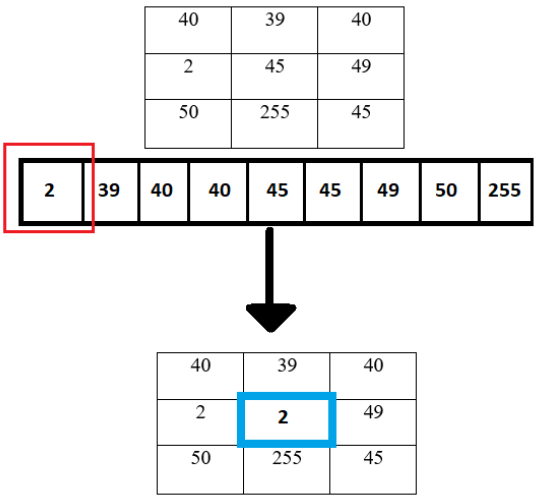


Ilustración 22. Representación de filtro mínimo

- Filtro del máximo: que selecciona el valor más alto del conjunto de pixeles para que el píxel seleccionado tome ese nuevo valor.

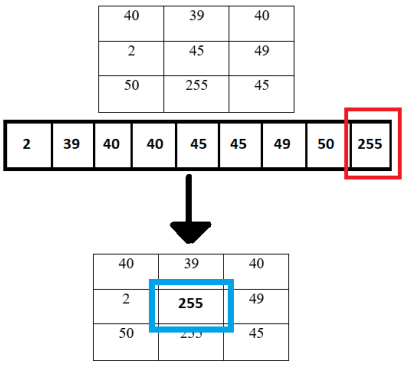


Ilustración 23. Representación de filtro máximo

- Filtro de la mediana: selecciona el valor intermedio del conjunto de pixeles.

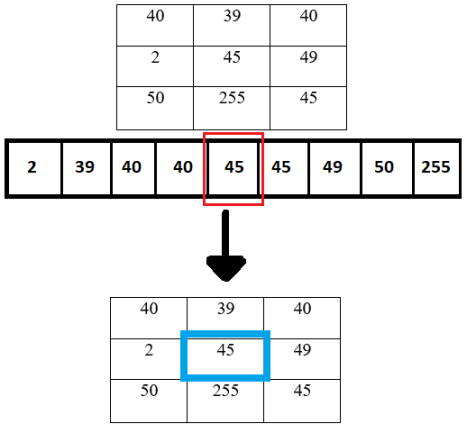
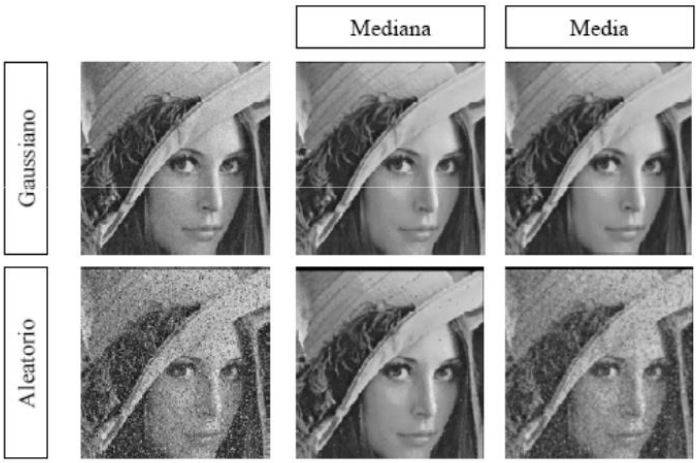


Ilustración 24. Representación de filtro mediana

En nuestro caso, queremos eliminar a toda costa el ruido impulsional, que se traduce como saturaciones del color blanco y negro, que son el valor máximo y mínimo que puede tomar un pixel en una imagen. Los dos primeros filtros lo único que harían sería amplificar el ruido impulsional, en el sentido de que ahora no serían unos pixeles aislados que se saturan, sino que se saturarían un conjunto de estos.

Es el filtro de la mediana el que se debe usar, pues tomando el valor intermedio del conjunto de pixeles, sabemos que ni el máximo (blanco) ni el mínimo (negro) son los valores finales para el pixel aislado. Con este filtro eliminamos completamente el ruido impulsional, como se muestra en la siguiente imagen:



Mateu Villa (2015). Ilustración 25. Comparativa de distintos filtros para eliminar ruido. Recuperado de http://www.ugto.mx/

En nuestro caso, conseguiremos eliminar los defectos producidos en la imagen por los rayos solares, tanto reflejos de la luz solar como sombras, haciendo que la región de color se comporte de una manera homogénea. Con esto conseguimos una diferenciación de las celdas respecto al terreno, que en este caso es de césped:

Ilustración 26. Resultado tras aplicar el filtro de la mediana

3.3 Preprocesamiento. Técnicas 'Opening' y 'Closing'. Operaciones morfológicas.

Hasta ahora, hemos estado realizando tratamientos en la imagen para mejorar las características de los datos que queremos estudiar y realzarlos frente a aquellos datos que no necesitamos. Los datos que queremos aislar frente al resto de la imagen serán objetos que queremos detectar. Se caracterizan por estar bien diferenciados entre sí y con sus bordes bien marcados. Estos pasos previos a la identificación de dichos objetos se denomina pertenecen a la etapa de preprocesamiento de una imagen.

Una vez tenemos las características realzadas de los objetos que queremos analizar, procedemos a su separación respecto del resto de la imagen. Esta separación pertenece a la etapa de segmentación de la imagen.

La segmentación de una imagen consiste en dividir una imagen digital en zonas individualizadas, con el objetivo de diferencial los distintos objetos que presentan. Al final de la etapa de segmentación, se debe saber a la perfección los objetos que hay para extraer las características propias de cada uno. La detección de dichos objetos se representa en forma de imagen binaria.

Una imagen binaria representa los objetos que queremos analizar (en color blanco), frente a los datos que queremos prescindir, considerados como fondo (color negro).En la mayoría de las figuras anteriores, se enseñaba tanto la imagen tratada como su imagen binaria resultante para facilitar la comprensión de lo que se estaba realizando al aplicar un filtro.

En nuestro caso práctico, procedemos al proceso de segmentación realizando una separación de los colores más cercanos al blanco (para separar las celdas, que son de color blanco), respecto del resto de colores de la imagen. Así, obtenemos la imagen de la Ilustración 26.

Se puede intuir la importancia que tiene hacer un preprocesamiento en la imagen, antes de detectar los objetos deseados, debido a que el ruido provocaba la detección errónea de objetos que no necesitábamos analizar.

Tras volver a ver la figura anterior, podemos observar cómo se ha reducido considerablemente el número de objetos en la imagen binaria, a consecuencia de aplicar el filtro de la mediana en la imagen, pero sigue habiendo pequeños objetos que no han sido posible eliminarlos por este filtrado. Para poder eliminarlos, tendremos que hacer uso de otro tipo de filtro que permita conservar el objeto referente a las celdas del terreno (incluso mejorar su identificación), además de eliminar por completo los objetos detectados por el ruido restante.

La principal diferencia respecto al filtro de la mediana, es que en este nuevo filtrado se trabaja directamente sobre la imagen binaria.

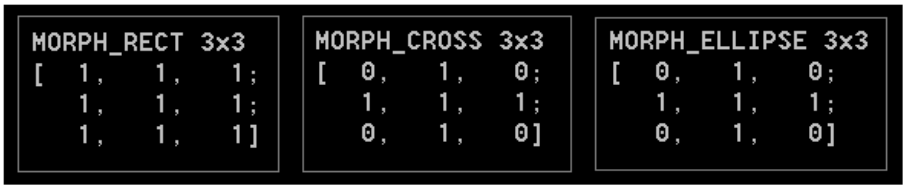
Al realizar modificaciones en una imagen binaria, estaremos hablando operaciones morfológicas, que tienen como objetivo obtener una nueva imagen binaria solución con los objetos identificados de una manera mejor. Una operación morfológica es aquella herramienta matemática utilizada para simplificar los datos en una imagen, en el sentido de conservar las características esenciales y eliminar aquellos datos que nos son irrelevantes. Es una muy buena herramienta a utilizar como paso previo al procesamiento real de los objetos. En nuestro caso dicho procesamiento real sería la clasificación y detección de las celdas.

Realizamos estas transformaciones morfológicas para cambiar la forma y estructura de los objetos. Con estas herramientas podemos obtener componentes que dan una idea de la forma y estructura de los objetos deseados. Además, nos permite la detección de contornos primarios, incluso obtener contornos que están dentro de un entorno con mucho ruido, como nuestro caso.

Dentro de las transformaciones morfológicas, nos centraremos en las que hemos utilizado en el código: la dilatación y la erosión de la imagen.

Para poder realizar este tipo de transformaciones necesitamos la imagen binaria a tratar y una estructura morfológica, que se usará como mascara en la operación matemática. La máscara se aplica en la imagen binaria, y tras realizar la operación morfológica, obtenemos una imagen binaria solución.

Dichas estructuras morfológicas pueden tener una forma predefinida o podemos personalizar la nuestra propia. En la siguiente imagen podemos ver un ejemplo de tres mascaras predefinidas:



(2017). Ilustración 27. Estructuras morfológicas. Recuperado de http://acodigo.blogspot.com

Cuanta más grande sea la máscara, mayor será la transformación del objeto, por lo que depende del caso de estudio tendremos que utilizar una máscara mayor o menor. En nuestra aplicación, utilizamos una máscara rectangular de 20x20 píxeles.

La dilatación en una imagen binaria consistiría en: dada una máscara B y una imagen A, el conjunto de desplazamiento de x tales que B y A se solapen en al menos un elemento distinto de cero.

En otras palabras, la dilatación consiste en pasar la máscara por cada uno de los pixeles de la imagen, y si alguno de los elementos de la máscara coincide con un pixel de la imagen, entonces el pixel de la imagen se pone a 1, y pasa a considerarse como parte del objeto. Veamos un ejemplo práctico:

Imaginemos que tenemos una máscara B:

0	1	0
1	1	1
0	1	0

Ilustración 28. Mascara B.

Y una imagen A:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Ilustración 29. Representación de una imagen binaria.

La dilatación consiste en pasar la máscara B en cada uno de los pixeles de la imagen A, y si uno de los elementos de la máscara coincide con un valor del pixel de la imagen, dicho pixel se pone a valor 1:

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Ilustración 30. Imagen solución tras aplicar dilatación

Por otro lado, el proceso de erosión consistiría en pasar una máscara B en una imagen binaria A, y ver si todos los puntos de la máscara que estén en valor lógico ‘1’ coincidan con los de la imagen, que se ponen a 1. En caso contrario se pone a 0. De nuevo, haremos uso de un ejemplo para entender mejor el concepto:

Sea una imagen A:

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

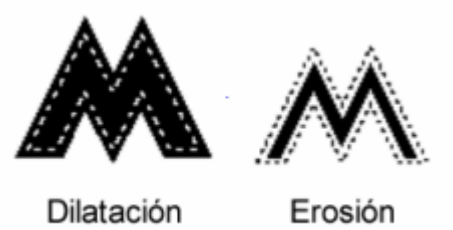
Ilustración 31. Representación de una imagen binaria.

Y una máscara B, como la de la Ilustración 28, el proceso de erosión consistiría en comparar los elementos a 1 de la máscara con los pixeles a 1 de la imagen, y si uno de ellos no coincide poner a 0 el pixel de la imagen. El resultado tras aplicar el proceso de erosión sería:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Ilustración 32. Imagen solución tras aplicar erosión

En términos todavía más sencillos, y para terminar de explicar la utilizad de estas dos operaciones morfológicas, definimos como proceso de dilatación de una imagen binaria aquella que aumenta las dimensiones del objeto en sus bordes, mientras que la erosión produce el efecto contrario: reduce las dimensiones del objeto en sus bordes. Se puede ver esta explicación en la siguiente imagen, donde el contorno formando por puntos suspensivos se corresponde con la imagen original, antes de la transformación morfológica:



(2016). Ilustración 33. Comparativas entre dilatación y erosión. Recuperado de https://slideplayer.es/slide/91852/

En nuestro código, utilizamos estas dos propiedades que tienen estas herramientas matemáticas, la de aumentar o disminuir las fronteras, para solucionar un problema que tenemos a raíz del ruido de la imagen. Sabiendo que si realizamos un proceso de dilatación en un objeto de la imagen, y posteriormente una erosión, ambos procesos con la misma estructura morfológica, obtenemos como resultado el mismo objeto (y con el caso contrario, erosión y luego dilatación, pasaría exactamente lo mismo), podemos realizar combinaciones de ambas operaciones para eliminar errores e identificar correctamente los objetos deseados.

Imaginemos que tenemos un objeto identificado por error que queremos eliminar. Dicho objeto es pequeño, pero es necesario eliminarlo para no alterar el código. Podemos realizar una operación de erosión de tal manera que elimine dicho objeto, pero a la vez no perjudique al objeto que esté identificado de manera correcta. Después de eliminar dicho objeto por el proceso de erosión, podríamos realizar un proceso de dilatación para que el objeto que queramos identificar de manera correcta recupere su forma original. A este proceso se le denomina apertura morfológica (Erosión + dilatación). En el siguiente flujo se muestra su funcionamiento:

Partimos de una imagen con un objeto identificado correctamente más objetos que se han detectado por error:

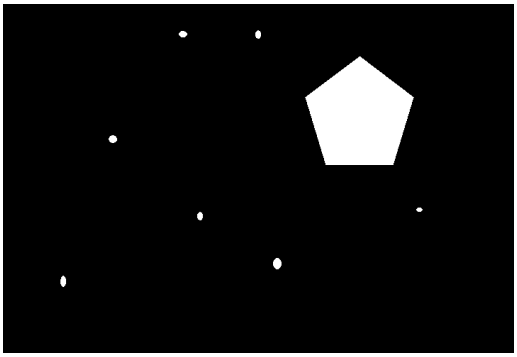


Ilustración 34. Imagen binaria con pequeños objetos de ruido

Al aplicar una erosión en la imagen, obtenemos el siguiente resultado:

Ilustración 35. Imagen solución con dimensiones incorrectas.

Conseguimos eliminar los objetos erróneos de la imagen, pero ahora el objeto deseado ha reducido su contorno, para arreglarlo realizamos una dilatación con la misma mascara estructural:

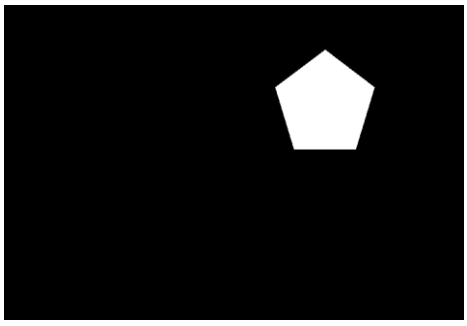


Ilustración 36. Imagen solución con dimensiones correctas.

Si en vez de tener objetos identificados por error, tenemos el objeto deseado identificado de una manera incorrecta, en el sentido de que tenemos parte de fondo dentro del propio objeto que queramos rellenar, podemos hacer uso de la operación morfológica de Cierre. Esta consistiría en dilatar primero la imagen, rellenando ese espacio vacío del objeto, para luego erosionar el objeto y obtener su contorno original, pero sin fondo dentro de este. De nuevo, se ejemplifica con las siguientes imágenes:

Partiendo de una imagen A:

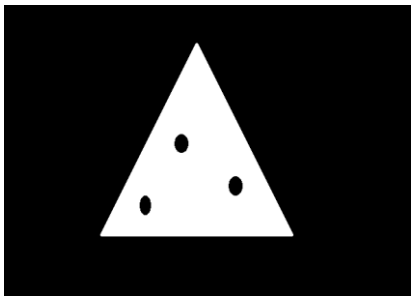


Ilustración 37. Imagen binaria con "agujeros fondo" dentro del objeto deseado.

Procedemos a realizar una dilatación para cubrir los “huecos” producidos por la mala detección del objeto, obteniendo la siguiente imagen:

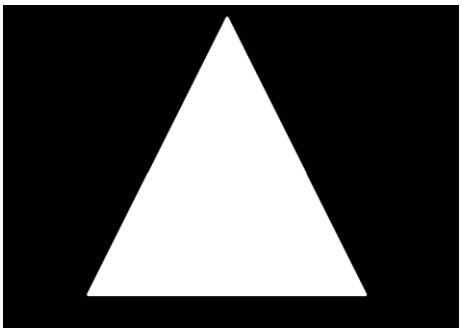


Ilustración 38. Imagen solución con sus dimensiones agrandadas

Los huecos identificados como fondo, pasarían a ser parte del objeto pero, a consecuencia de la dilatación, ahora el objeto tendría un área mayor. Para solucionarlo, erosionamos el objeto para que recupere su contorno original:

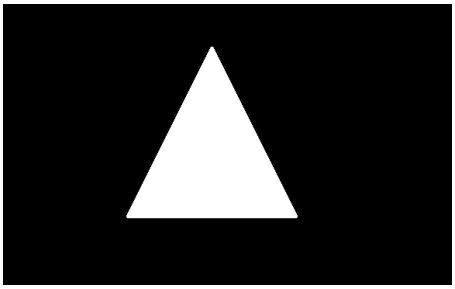


Ilustración 39. Imagen solución correcta.

En nuestro algoritmo, hemos realizado, en el mismo orden, un cierre y una apertura, obteniendo como resultado la correcta identificación de las celdas del terreno, como se aprecia en la siguiente imagen:

Ilustración 40. Imagen solución binaria final para nuestro caso de uso real.

3.4 Detección de bordes. Clasificador Canny. Detección de centros de las celdas.

El siguiente paso en nuestro código consistiría en la clasificación y ordenamiento de las celdas, en formato 'buscaminas' para poder tener organizado el terreno. De la figura anterior, tenemos que ser capaces de extraer el centro de cada celda y poder ordenarlas por filas y columnas. Para ello, debemos encontrar primero el contorno de cada celda, y a partir de ahí identificar el centro de cada contorno.

El problema que tenemos ahora es que los bordes de las celdas, que son parte del objeto de la imagen, son demasiado anchos. Esto provoca que para cada celda, tengamos bastantes contornos con sus respectivos centros, por lo que añade mucho ruido a la hora de detectar un solo centro.

Una posible solución sería extraer los bordes del objeto (las celdas del terreno, en la imagen binaria), con lo que se reduce considerablemente el número de contornos detectados.

Antes de extraer los bordes de la imagen, debemos saber lo que es. Un borde es un cambio brusco de valor entre dos píxeles vecinos. La siguiente gráfica de función representa un borde, donde el cambio de 0 a 1 ejemplifica mejor su representación:

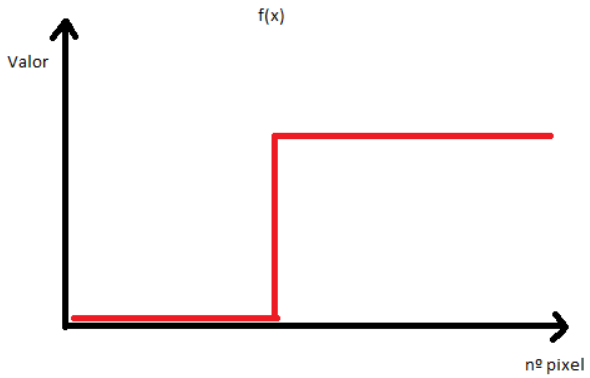


Ilustración 41. Representación gráfica de una función borde.

El cambio brusco de valor se identificaría como borde. Una manera de identificar dicho borde sería haciendo uso de la primera derivada, donde obtendríamos el máximo de la función en la zona donde se cambia bruscamente el valor de la función:

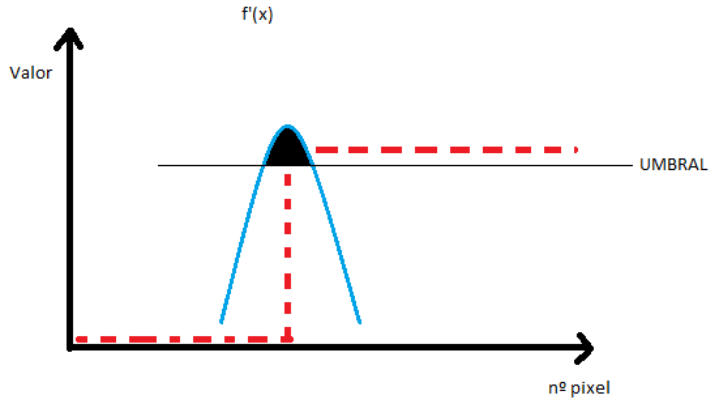


Ilustración 42. Función primera derivada de la función borde.

Realizando operaciones matemáticas sobre la imagen donde se haga uso de la primera derivada, obtenemos los valores máximos de la función resultado en la zona donde cambia tan bruscamente la función original. Para diferenciar el borde de lo que no es borde, tendremos que establecer una tolerancia, en forma de umbral para la función, que filtre lo que se va a considerar borde y lo que no. Para saber el punto exacto donde está el borde, tendremos que hacer operaciones que conlleven el uso de la segunda derivada, y es el punto de inflexión de la función resultado donde se sitúa el borde del objeto. En la siguiente imagen vemos el resultado de aplicar la segunda derivada a la imagen para detectar el borde:

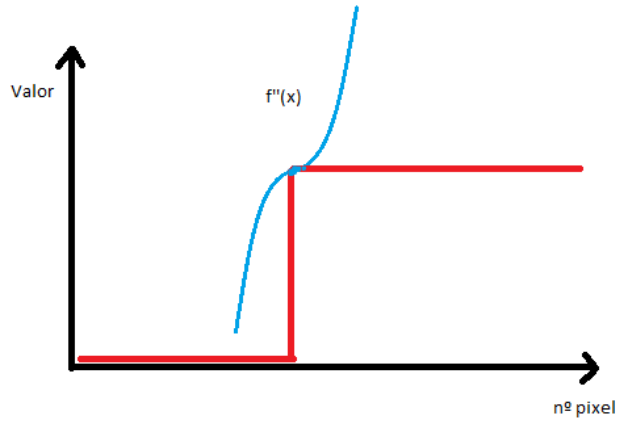


Ilustración 43. Representación de la segunda derivada de una función borde.

La base de la detección de los bordes en una imagen consistiría en aplicar estas derivadas al proceso de detección. Hay métodos más complejos, con los que conseguimos resultados del estilo detección de bordes de un solo pixel de ancho. En nuestro caso, es lo que queremos, para reducir al mínimo el número de contornos detectados. Para ello, vamos a hacer uso de un algoritmo de detección de bordes, denominado el Algoritmo de Canny.

El algoritmo de Canny hace honor a su creador, John F. Canny, que en 1986 desarrolló el algoritmo con el objetivo de identificar los contornos en una imagen. El algoritmo parte de tres premisas que se deben cumplir:

- La premisa de detección indica que no debe haber ningún borde falso, y que los bordes importantes no deben estar eliminados. En nuestro caso práctico, se refiere a que debemos eliminar aquellos objetos que solo añaden ruido a la imagen y además el borde del objeto debe encajar a la perfección con el objeto real a analizar.
- La premisa de la localización establece que la distancia entre la posición real del borde del objeto y la localizada del borde debe ser mínima. Para ello, debemos asegurarnos la correcta separación del objeto respecto al fondo, obteniendo la imagen binaria limpia.
- Por último, la premisa de la mínima respuesta obliga al algoritmo a dar como solución un único borde, en el sentido de eliminar todo aquel borde repetido a causa del ruido de la imagen, además de eliminar aquellos falsos bordes.

Una vez enunciadas los tres criterios primordiales para el uso del algoritmo, procedemos a su explicación y funcionamiento por etapas. El operador de Canny consta de 3 fases, si bien algunos autores hacen uso de una cuarta fase, que incluiremos en la explicación. Estas son:

- Obtención del Gradiente: Aquí aparece parte de la teoría explicada anteriormente para la detección de bordes, pues el gradiente de una imagen bidimensional consistiría en la aplicación de las derivadas parciales sobre cada eje de la imagen.

  Antes de calcular el gradiente de la imagen, se requiere pasar un filtro para eliminar los posibles ruidos presentes en la imagen. En nuestro caso, ya lo hemos realizado.

  El gradiente de una imagen f(x,y) en un punto (x,y) se define como un vector bidimensional dado por la ecuación:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

(2016). Ilustración 44. Formula de Gradiente. Recuperado https://slideplayer.es/slide/1737315/

  Para cada pixel obtenemos la magnitud y modulo (orientación en grados) del gradiente, por lo que si tenemos dos ejes en la imagen, obtenemos dos imágenes. Para obtener solo una imagen solución con los bordes detectados, pasamos a la siguiente etapa del algoritmo de Canny.

- Supresión no máxima al resultado del gradiente: Con las dos imágenes generadas en el paso anterior, obtendremos una imagen con los bordes adelgazados, del tamaño de un pixel de grosor. El procedimiento de solapamiento de imágenes consistiría en lo siguiente: identificamos cuatro orientaciones respecto al eje horizontal (0º,45º,90º y 135º). Para cada uno de los pixeles se encuentra la dirección que mejor se ajuste a la dirección del Angulo del gradiente. Después, observamos si el valor de la magnitud del gradiente para esa dirección es más pequeño que al menos uno de sus vecinos. En caso de ser así, el pixel pasa a tener valor 0, y en caso contrario se le asigna el valor de la magnitud del gradiente.

  Con esta etapa obtenemos una imagen binaria con los bordes detectados, todos ellos con un grosor de 1 pixel de ancho.

- Histéresis de umbral a la supresión no máxima: Como la imagen resultante del paso anterior tiende a tener máximos locales a causa del ruido, haremos uso de la histéresis del umbral.
  El proceso funciona tomando la imagen del paso anterior, la orientación de los puntos de borde de la imagen y escoger dos umbrales (superior e inferior, T2 y T1). En cada punto de la imagen se debe cumplir que el siguiente punto del borde no explorado sea mayor al segundo umbral (P(x,y) >T2). A partir de ese punto, encadenar los máximos locales en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral (que era más pequeño que el segundo, T1<T2). Vamos marcando los puntos que cumplen el criterio como puntos explorados, que pasan a formar parte del borde.
  Con este paso conseguimos eliminar las uniones “en forma de Y” se los segmentos que se unen en un único punto.

- Cierre de contornos abiertos: Esta última etapa, consiste en cerrar aquellos contornos que puedan haber quedado abiertos por el ruido de la imagen. Un método muy utilizado en este paso consiste en la aplicación del algoritmo de Deriche y Cocquerez. Dicho algoritmo hace uso de una imagen binaria de entrada, con contornos de un pixel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con el otro extremo.

Cada vez que utilizamos el Algoritmo de detección de bordes de Canny, realizamos estas cuatro etapas para su correcta aplicación. El resultado final consiste en una detección de contornos de un pixel de ancho, del objeto que queremos analizar. En nuestro caso, obtenemos la siguiente imagen binaria al aplicar el filtro de Canny:

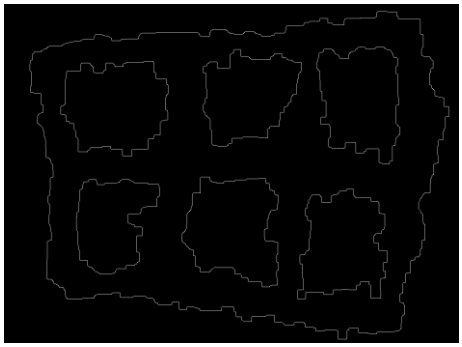


Ilustración 45. Solución final tras aplicar el detector de bordes Canny.

Se puede comprobar la eficacia del Operador Canny, que en gran medida se ha debido a un correcto preprocesamiento de la imagen binaria (eliminación de ruido por operaciones morfológicas). Se puede observar además que obtenemos dos contornos para cada celda, uno exterior y uno interior. Esto se debe a que el grosor de las líneas de la celda era significativamente anchos, por lo que tienen dos contornos (exterior e interior). Para nuestra solución, solo utilizaremos los interiores para la detección del centro de la celda. La imagen resultante tras eliminar los contornos exteriores sería la siguiente:

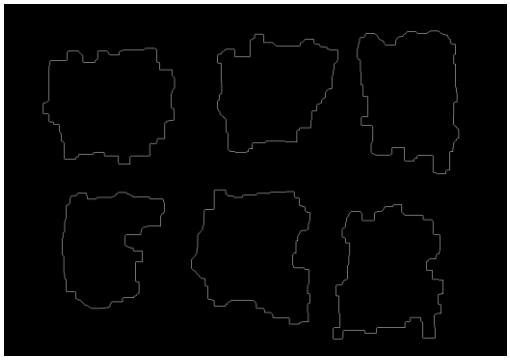


Ilustración 46. Imagen solución binaria sin los bordes exteriores.

Una vez detectados los contornos de las celdas, procedemos a la detección de sus centros. Como estamos tratando con terrenos irregulares, los contornos de las celdas también serán irregulares, en vez de rectángulos bien definidos. Esto afecta en gran medida a la detección del centro, que estamos intentando detectar para ofrecer más datos sobre la posición de la celda al dron. En realidad, no necesitamos saber la posición exacta del centro, sino una aproximación. Esto se debe a que sabemos que cada celda, aun siendo completamente irregular para la imagen, va a tener la misma área. Sabiendo esta condición, si detectamos un punto como centro dentro del área, podemos posicionar la celda respecto a un punto origen y mandarle correctamente las coordenadas al dron. Para detectar dicho centro, vamos a aproximar cada contorno irregular a un rectángulo que contenga cada pixel del contorno y a la vez se ajuste lo más posible. El centro del rectángulo resultante será el centro del contorno.

El procedimiento se refleja en la siguiente figura, donde se muestra el centro del rectángulo (en rojo) y se estima el centro del contorno (en blanco):

Ilustración 47. Imagen donde se representa la detección del centro del rectángulo equivalente a una celda, y la estimación del centro real.

3.5 Ordenamiento de las celdas. Representación matricial

Como se ha explicado antes, detectamos los centros de las celdas para que el dron tenga una referencia en caso de que quiera examinar una celda determinada, para su detección de mina. Sabiendo un punto de origen y que cada celda mide 1 metro cuadrado de área, podemos enviar al dron a la celda que queramos, solo si somos capaces de organizarlas por filas y columnas en un array.

En la siguiente foto tenemos los centros de las celdas detectados:

Ilustración 48. Imagen con los centros detectados.

Para organizar las celdas por filas y columnas de forma virtual, lo mejor es almacenar los centros en un array de 2 filas x 3 columnas. Este array representa de manera virtual el terreno con celdas.

En este proceso de guardado de centros, nos encontramos con el siguiente problema. Si bien la manera más intuitiva para guardar los centros es la de seguir el mismo orden de las celdas (recorrer un bucle en toda la imagen que vaya de izquierda a derecha y de abajo a arriba), nos encontramos con que la posición de los centros altera la posición de guardado en el array, obteniendo el siguiente orden:

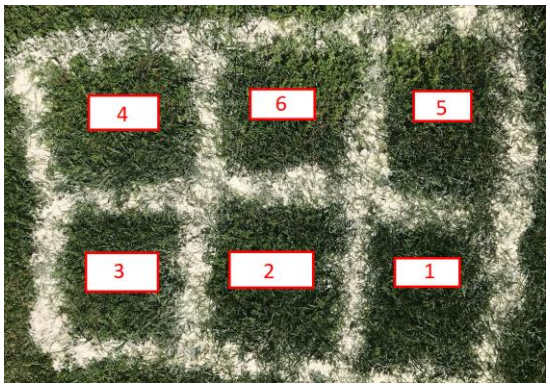


Ilustración 49. Imagen con los centros desordenados.

Esto se debe a que se está tomando como cierto que los centros están alineados en la misma línea horizontal, y al no cumplirse esta condición por tener las celdas formas irregulares, se produce el error al utilizar ese método de ordenamiento. Para solucionarlo, basta con ir agrupando centros de tres en tres (de abajo a arriba) e ir añadiendo cada centro tomando como orden de izquierda a derecha. Así, vamos rellenando cada fila de celdas con el orden correcto para cada celda, obteniendo un array con los centros en su posición de array correcta, como se muestra en la siguiente figura:

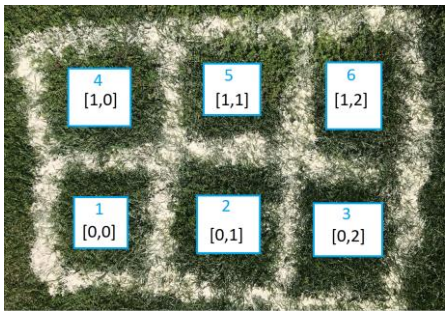


Ilustración 50. Imagen con los centros ordenados.

Una vez bien posicionadas y detectadas las celdas del terreno, podemos proceder a enviar la información al resto de dispositivos del proyecto para poder localizar las minas y ofrecer una solución para su desactivación.

Dentro de la programación del dron, este debe reconocer tanto las celdas del terreno irregular como las minas contenidas en ella, y debe ser capaz de enviar la información sobre la localización de dichas minas y celdas a los demás dispositivos del proyecto (Rover, Pc en tierra,…).

Este Trabajo de Fin de Grado se centra en la comunicación interna del dron y sus distintos fragmentos de código, para ofrecerle el array de celdas como información y que se nos devuelva como respuesta la posición de la mina o minas contenidas en ellas. Para poder enviar el array de celdas posicionadas, debemos hacer uso de algún programa que permita ejecutar varios códigos a la vez, puesto que el dron va a tener que

detectar las celdas y a la vez posicionarse sobre cada una de ellas para detectar posibles minas, y a la vez poder enviarse parámetros desde un código a otro, sin tener que pausar la ejecución de ninguno.

La solución a estas premisas las encontramos haciendo uso del Sistema Operativo Robótico (o sus siglas en inglés, ROS).

3.6 Envío de centros ordenados y recepción de celdas con minas. ROS.

Haciendo mención a la página oficial, ROS es -"*Un framework flexible para desarrollar software dedicado a robots*"-. Es una colección de librerías que nos permite simplificar tareas que normalmente requerirían de mucha complejidad, al tratar de desarrollar una programación robusta y complicada, como son la de los robots.

El gran potencial que tiene esta herramienta es que podemos dividir en distintas tareas diferentes funciones que tenga que desempeñar el sistema embebido. Se puede explicar mejor con un ejemplo: Si estamos desarrollando un dron autónomo, un grupo se puede dedicar al desarrollo de software relacionado con el control de vuelo y dirección, mientras que otro se puede dedicar a la detección y reconocimiento de rutas de vuelo. Podemos fraccionar distintas tareas del sistema embebido y comunicarlas entre sí haciendo uso de ROS.

Estos fragmentos de código se comunican entre sí, y la idea general de ROS es que estos fragmentos de códigos se ejecutan a la vez, y toman la información que necesiten de otros códigos sin tener que pausar la ejecución del programa.

La estructura general de un programa en ROS sigue el siguiente esquema:

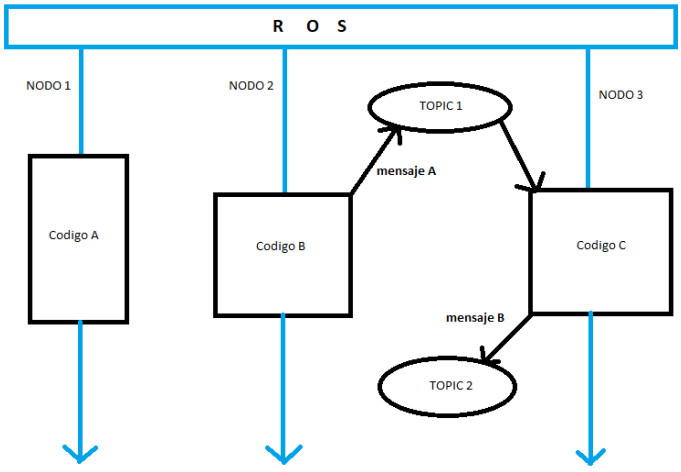


Ilustración 51. Esquema de funcionamiento del Sistema Operativo Robótico (ROS, por sus siglas en ingles).

Podemos distinguir, a raíz del esquema, 4 elementos principales para la ejecución de ROS: Nodos, Topics, Mensajes y Códigos. Los códigos son los fragmentos de código que se dividen para abarcar distintas tareas que tiene el robot.

Cada programa de ROS tiene identificados los fragmentos de código encapsulándolos en nodos. Cada nodo tiene una función a desempeñar distinta del resto de nodos, y por tanto encapsulará un fragmento de código específico para cada función a desempeñar. Los nodos pueden ejecutarse sin la necesidad de comunicarse con otros nodos, o pueden hacer uso de parámetros provenientes de otros nodos usando como conexión los Topics.

Un topic es el espacio donde se deposita información útil para otros nodos. Todo nodo que necesite un parámetro para desarrollar el código de manera correcta, hará uso de un topic para recoger dicha información. La información o parámetros se envían en forma de mensaje. Dicho mensaje puede abarcar cualquier estructura, desde un mensaje de tipo String hasta un mensaje de tipo estructura, que contenga distintas variables.

En nuestro algoritmo, hacemos uso de la metodología Publisher/Subscriber para pasar y recibir información de un código a otro. Un nodo actúa como Publisher si envía a un topic información en forma de mensaje, mientras que un nodo actúa como Subscriber si recibe información de un topic para la ejecución del código.

Se puede intuir lo que vamos a pasar como información, a raíz de apartados anteriores, al dron y al resto de dispositivos: las celdas ordenadas por filas y columnas. Para ello, hacemos uso de un mensaje de tipo String, que tendrá la siguiente estructura:

X-Y-A-B-0

Ilustración 52. Representación de un mensaje genérico con las coordenadas de cada centro.

- Los dos primeros componentes que conforman el mensaje, X e Y, son las coordenadas del centro de la celda en la imagen, medidas en pixeles. Esta información es útil para posicionar el dron, haciendo uso de lógica, en la celda en la que queramos verificar si hay minas o no.
- Los dos siguientes componentes del mensaje, A y B, hacen referencia a la posición dentro del Array. Equivaldría a la posición en el formato 'buscaminas'.
- El ultimo componente del mensaje, el 0, se manda en el propio mensaje para facilitar la lectura de este. Se refiere al número de minas detectado en la celda. Como todavía no se han procesado los datos de las celdas, no sabemos las posiciones de las minas.

Cada celda tiene su mensaje propio, que se publica en el topic del Publisher siguiendo el orden de las celdas.

Posteriormente, en el mismo código, actuaremos como un Subsriber de un nuevo Topic para recibir la posición y numero de minas en cada celda, y poder desarrollar una solución que implique desactivar las minas si hacer explotar ninguna.

Nos subscribimos al nuevo topic y vamos rellenando, en el mismo orden de llegada, el nuevo array que representa las celdas con las minas contenidas en ellas.

Independientemente del número de minas, si en una casilla se detecta una mina, dicha celda se marcaría para su desactivación de forma segura mediante el uso del vehículo autónomo Rover.

Así, independientemente del número de minas en una celda, podemos representar de manera gráfica el aspecto que va a tener una aplicación real del algoritmo:

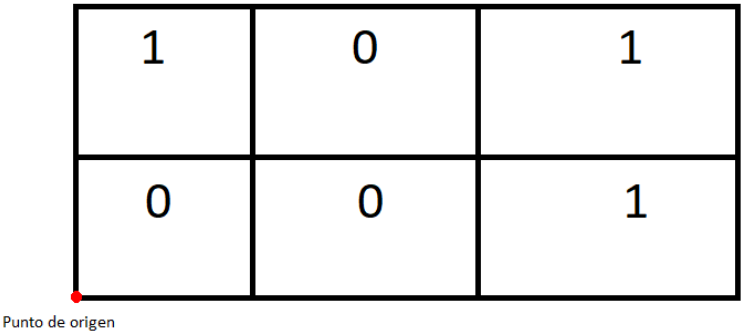


Ilustración 53. Representación de un terreno con minas en algunas de sus celdas.

Donde las casillas marcadas con un '1' lógico representan casillas con minas en ellas, mientras que las casillas marcadas con un '0' lógico representan casillas que son seguras para visitar. Será el vehículo autónomo el encargado de seguir estas rutas para desactivar las minas.

3.7 Generación de rutas solución para Vehículo autónomo terrestre (Rover).

Antes de proceder a explicar el algoritmo que genere dichas rutas solución, tenemos que resumir las premisas o condiciones para generar las rutas:

- No se puede visitar ninguna celda que contenga minas en ella. El Rover solo puede visitar celdas seguras de minas.
- El Rover desactiva las minas desde una posición segura. Esto quiere decir que el destino para desactivar una casilla con minas, será una celda adyacente vecina a dicha celda (celdas vecinas verticales u horizontales). A estas celdas adyacentes se conocen con el nombre de celda segura
- Toda casilla con mina tiene que tener una casilla segura accesible para poder realizar satisfactoriamente el proceso de desactivación. En caso de no tener ninguna casilla segura desde la que se pueda desactivar la mina, el propio programa manda un mensaje de "error al crear ruta".
- No se puede generar ninguna ruta que contengan puntos externos a las celdas. Siempre se mueven por estas.

Una vez explicadas las premisas, podemos buscar el mejor método para crear rutas seguras para el vehículo autónomo.

Entre todos los métodos de creación de rutas, destaco el que más se ajusta a nuestra aplicación, y ese método de resolución se denomina metodología backtracking.

La estrategia de Backtracking (‘Vuelta atrás’) se utiliza para encontrar soluciones a problemas que contienen restricciones. El término fue utilizado por primera vez en la década de 1950 por el matemático Derick Henry Lemer.

La idea de la metodología backtracking es encontrar la mejor combinación posible a un problema determinado. Durante la búsqueda, se estudia las distintas posibilidades de la solución, y si alguna posibilidad muestra problemas o da fallo, se vuelve al paso anterior para probar con otra vía posible. Si no hay más combinaciones posibles, el algoritmo falla. En sí, el algoritmo crea un árbol de búsqueda con todas las alternativas posibles, indicando cuales son posibles soluciones y cuales son alternativas incorrectas. Veamos un ejemplo para entender mejor el algoritmo. Imaginemos una matriz 4x4 con casillas que no se pueden visitar, donde se quiere partir de un punto A para llegar a un punto B, como en la siguiente figura:

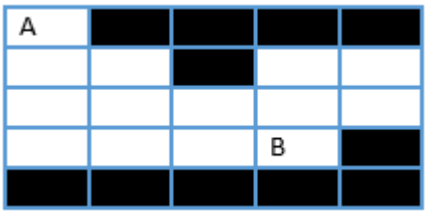


Ilustración 54. Matriz con celdas que no se pueden visitar pintadas en negro.

Las casillas en negro representan casillas que no se pueden visitar. Queremos estudiar la mejor combinación posible para movernos de un punto ‘A’ a un punto ‘B’, generando un árbol de búsqueda profunda mediante la metodología backtracking, en la que se consiguen todas las soluciones posibles para llegar al destino. El árbol tendría la siguiente forma

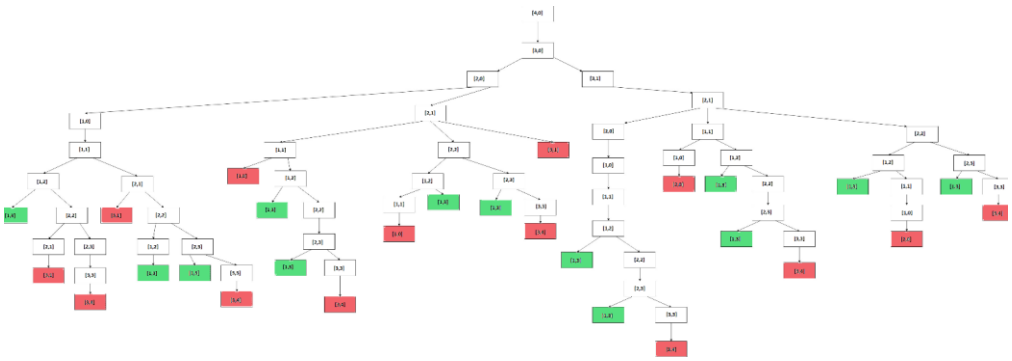


Ilustración 55. Árbol de búsqueda con todas las rutas solución posibles, en verde, y rutas sin solución, en rojo.

El árbol se desarrolla partiendo desde el punto A ([4,0]) hasta el punto B ([1,3]). Las casillas marcadas en rojo son rutas que no llegan al destino final, y por tanto hace uso de la metodología backtracking para volver a una bifurcación y probar por un nuevo camino. Las casillas en verde son rutas que han llegado correctamente a su destino.

Lo bueno que tiene este método es que solo se puede visitar la celda una vez, por lo que nos aseguramos que las rutas solución van a ser las más óptimas.

Para aplicar la metodología backtracking a nuestro código, tenemos que saber que las casillas que no podemos visitar son aquellas que contienen minas. Para su desactivación, tenemos que buscar primero una celda vecina adyacente a la que contiene minas y que a la vez sea posible llegar a ella, conformando una ruta segura para la desactivación de la mina.

Se crea una ruta solución para cada celda con minas, partiendo siempre desde la celda origen de coordenadas. Así, imaginemos que tenemos el siguiente terreno con minas:

Ilustración 56. Representación del terreno real con minas en algunas de sus celdas

Las celdas marcadas con una 'X' en rojo representan celdas con minas, mientras que la celda que contiene un cuadrado azul representa el origen de la ruta solución. Al tener tres celdas, se crearán tres rutas para la desactivación de las minas, obteniendo como resultado las siguientes rutas:

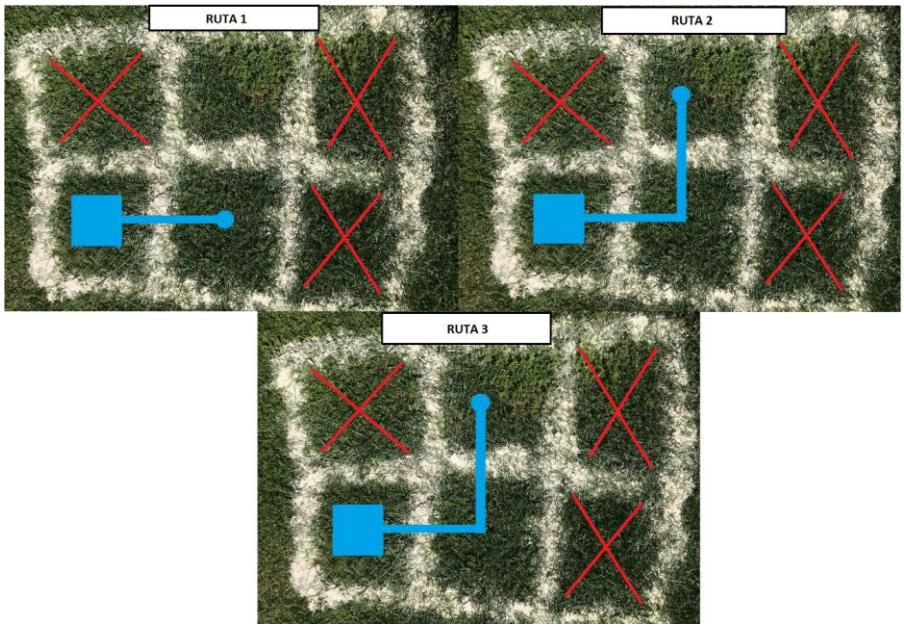


Ilustración 57. Representación de las rutas solución.

Se observa como coinciden dos rutas, debido a que se encuentran las minas en celdas accesibles paralelas. Dentro del código, la representación matricial de la ruta solución

añade un ‘1’ lógico a cada casilla visitada, mientras que el resto de casillas, aquellas que contienen minas y aquellas que no se han visitado, se incluyen con el valor lógico ‘0’. Tomando como ejemplo la ruta 3, tendríamos la siguiente matriz solución:

0	1	0
1	1	0

Ilustración 58. Representación matricial de una ruta solución.

Cada ruta se almacena en una matriz distinta. Como paso final del código, debemos componer el mensaje para ser enviado a un nuevo topic. Dicho mensaje sigue el orden del array, y va concatenando los valores incluidos en cada posición del array mediante guiones. Así, tomando la misma ruta 3 como ejemplo, obtendríamos el siguiente mensaje solución:

1-1-0-0-1-0

Ilustración 59. Representación del mensaje con la ruta solución a enviar.

De esta forma se buscaba la manera más sencilla de componer las rutas. El bucle recorre cada posición del array, siguiendo como orden Filas X Columnas, y va tomando cada valor de cada casilla. Una vez creado el mensaje, se envía a un nuevo topic para que se procesen los datos y se pueda enviar el vehículo autónomo a desactivar las minas. Para componer de nuevo la ruta solución, solo hace falta crearse un array vacío que represente el terreno con celdas, e ir leyendo cada uno de los valores del mensaje para ir rellenando las celdas con el valor correspondiente, en el mismo orden de Filas X Columnas.

4. RESULTADOS EXPERIMENTALES

4.1 Pruebas iniciales y aproximación al algoritmo solución

Al empezar a idear el algoritmo propuesto, no utilizaba imágenes de terrenos reales. En vez de eso, hice un boceto en un folio en el que dibujaba las celdas con una regla, quedando como resultado la siguiente imagen:

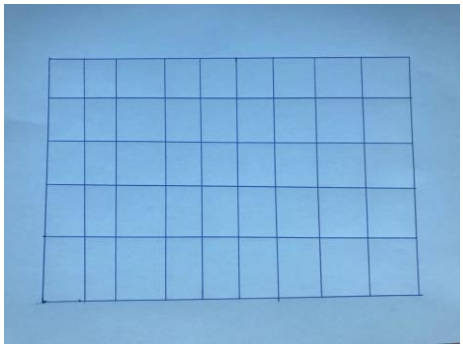


Ilustración 60. Imagen del boceto del terreno.

La idea era que el código que tuviera que escribir para resolver el terreno boceto no cambiara mucho con las imágenes de un terreno real. Lo que conseguía con esto era una mayor facilidad para interpretar datos y una claridad a la hora de aplicar filtros en el algoritmo, pues problemas de ruido tales como la luz solar no aparecían en el boceto. Así, con la utilización del boceto me centraba en conseguir detectar correctamente las celdas e identificar los centros de cada una, pues el envío de las celdas y la generación de rutas solución los resolvería con fotos de terrenos reales.

En un principio, seleccionando los valores de H, S y V apropiados para la detección de objetos inRange, era más que suficiente para separar como objeto, las celdas, respecto del resto de la imagen.

Aplicando el Operador de Canny para la detección de bordes, obteníamos los bordes exteriores e interiores de las líneas de las celdas, obteniendo como resultado la primera prueba real del algoritmo, aplicado a la detección de celdas:

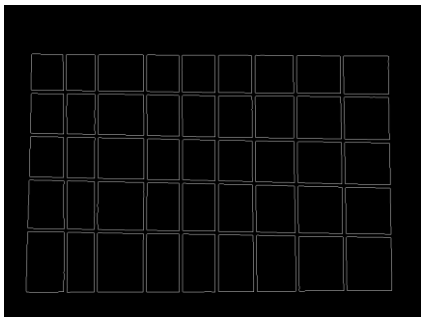


Ilustración 61. Detección de bordes de celdas del boceto.

Una vez reconocidas las celdas mediante el operador Canny, era muy sencillo detectar los centros de las celdas para una aproximación al ordenamiento por filas y columnas, por lo que realice una prueba de concepto rápida para marcar un centro de una celda con

un punto, y mostrarla por pantalla. El resultado final para la prueba de concepto sería el siguiente:

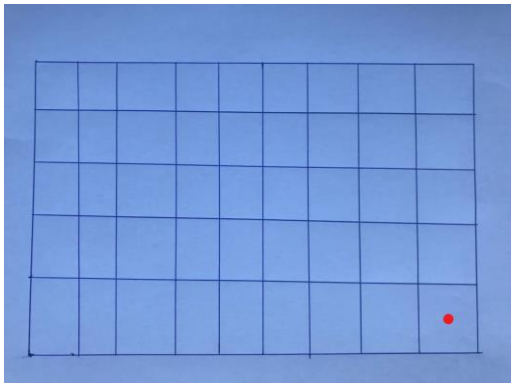


Ilustración 62. Detección de centro de celda, en rojo.

Este fue el primer resultado exitoso tras aplicar el algoritmo propuesto para la detección de celdas. A partir de la prueba concepto, no tendría que cambiar mucho el código, pues el filtrado se realizaría de una forma casi exacta. Así, una vez finalizado el experimento con el terreno boceto, podría centrarme en imágenes de terrenos reales.

4.2 Aplicación real del Algoritmo Propuesto para la detección de un terreno de una celda.

Al concluir con el éxito en la prueba del algoritmo, me propuse a hacer otro experimento para detectar correctamente un terreno real que contuviese solo una celda. El proposito del experimento era ver si el algoritmo utilizado en las pruebas de concepto era válido para detectar celdas en terrenos reales. Para ello utilicé la siguiente imagen:

Ilustración 63. Terreno real con 1 celda.

En un principio, me resultaba muy sencillo alterar los valores de H, S y V para que detectaran los colores blancos, obteniendo unos resultados bastante prometedores, reflejados en la siguiente imagen:

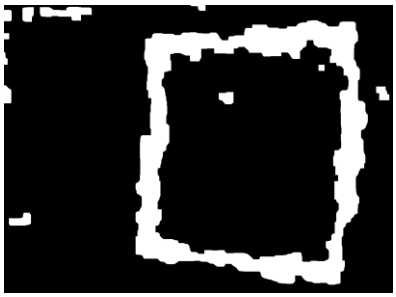


Ilustración 64. InRange del terreno real de una celda.

Ahora el problema que me encuentro es el ruido provocado por la luz solar. Aplicando un filtrado de mediana se solucionaba el algoritmo, por lo que el cambio respecto a la prueba anterior resultaba mínimo. El nuevo resultado limpiaría el ruido de la imagen binaria, y tras aplicar el operador Canny para reconocer los bordes de la celda, conseguiríamos la validez que tenía el algoritmo propuesto para las pruebas de concepto, que se muestran en la siguiente imagen:

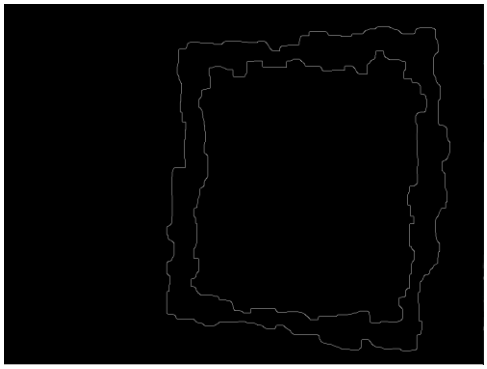


Ilustración 65. Detección de bordes del terreno real de una celda.

Finalmente, resultaba muy sencillo detectar el centro de la celda, por lo que de nuevo quedaba validado el experimento para el reconocimiento de las celdas en terrenos reales. El resultado final sería el siguiente:

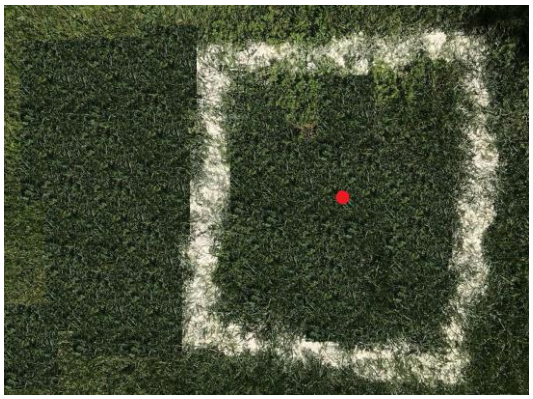


Ilustración 66. Detección de centro de celda, en rojo.

Quedaba validado así el algoritmo propuesto para la detección de centros en terrenos reales. Como última prueba, realizaría un experimento con un terreno real con varias celdas. Además, para esta última prueba generaría las rutas solución y las enviaría mediante ROSS a un topic solución, quedando validado por completo el algoritmo.

4.3 Prueba final para validación del Algoritmo Propuesto.

Para finalizar las pruebas del algoritmo propuesto, me dispuse a realizar una prueba final con una imagen real de un terreno con varias celdas. El objetivo era completar el código con la generación de rutas solución y el envío correcto de dichas rutas a un topic solución. La generación de rutas seguras se llevaba a cabo con una metodología famosa en el mundo de la programación, "*backtracking", por* lo que se podía utilizar la misma imagen alterando la posición de las minas para poder validar el algoritmo. Así, recurriría a la siguiente imagen final para validar del todo el algoritmo propuesto:

Ilustración 67. Imagen de terreno real con varias celdas.

4.3.1 Calibración y valores de HSV

Uno de los primeros problemas que me encuentro a la hora de realizar el filtrado de colores inRange, es que no sabía qué valores escoger de HSV para solo conservar aquellos colores pertenecientes a los bordes de las celdas. En un primer intento, me fui aproximando a base de prueba y error para conseguir diferenciar parte del terreno de celdas. Como no conseguía cuadrar los valores óptimos de HSV, investigué en internet y encontré un código de un usuario anónimo (Anexo 6.1) que mostraba unas barras de valores para los distintos valores H, S y V, respectivamente, y enseñaba en directo la imagen inRange resultante. La interfaz tiene la siguiente pinta:

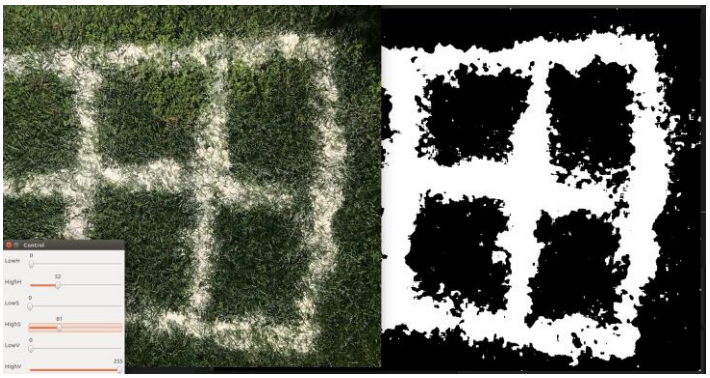


Ilustración 68. Interfaz del código HSV.

El código para seleccionar valores de HSV me agilizo mucho las pruebas, estableciendo como valores óptimos los siguientes: para H, 52; para S: 81; y para V: 255 (todos los valores tienen un rango de 0 a 255).

4.3.2 Preprocesamiento de la imagen

Posteriormente, me fije que la imagen resultante, tras aplicar el comando inRange, tenía mucho ruido e iba a producir errores en la detección de las celdas. A base de prueba y error, deduje que el problema era los efectos luminosos producidos por los reflejos de la luz solar directa. Para reducir al máximo el error producido, recurrí al filtro de la Mediana. Aquí se muestra una comparativa con los resultados:

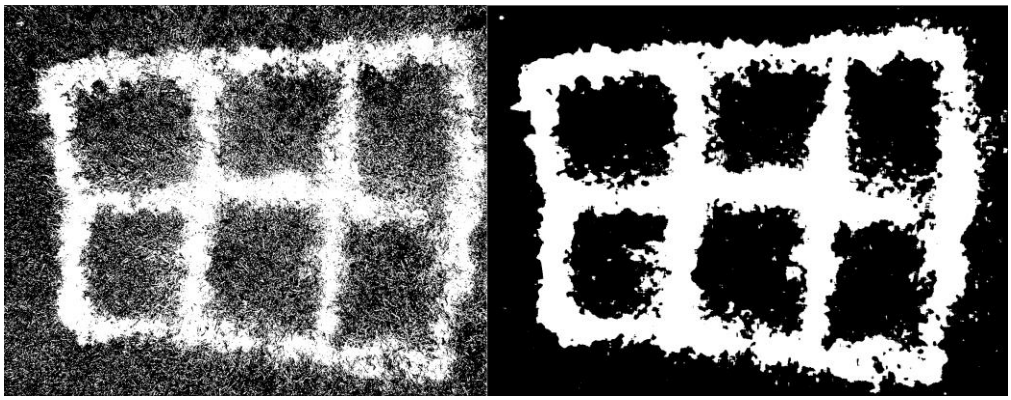


Ilustración 69. Comparativa sin y con filtro de la mediana aplicado.

El resultado de la imagen tras aplicar el filtro de la mediana reduce considerablemente el ruido producido por la luz solar, si bien siguen identificándose pequeños objetos que no se corresponden a las celdas del terreno. Para eliminarlo, recurrí a las técnicas de filtrado morfológico ‘Opening’ y ‘Closing’, obteniendo el preprocesamiento final mostrado en la siguiente figura:

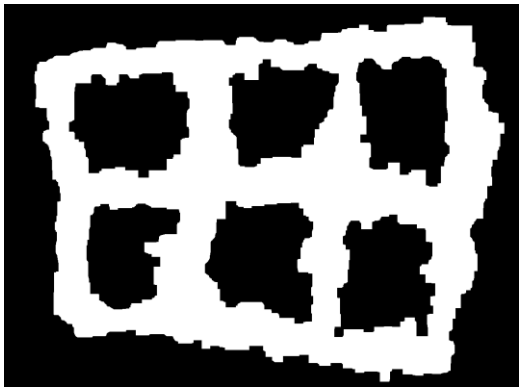


Ilustración 70. Resultado final tras preprocesamiento.

Ahora la imagen binaria quedaba perfecta para la extracción de las celdas, utilizando el algoritmo de detección de borde de Canny (los resultados del algoritmo quedan reflejados en el Capítulo 3). Una vez detectado los centros, el resultado sería el siguiente:

Ilustración 71. Centros detectados.

Donde cada punto de color rosa representa el centro de la celda detectada.

4.3.3 Composición de mensaje de envío. Envío celdas ordenadas.

Una vez detectadas las celdas, con sus centros, solo quedaba componer el mensaje String con las coordenadas de cada centro. Cada centro tiene dos tipos de coordenadas, su localización en la imagen y su posición en el array de celdas. Así, se componía el mensaje de tipo String y se enviaba al topic correspondiente. El envío exitoso de un mensaje queda registrado con un mensaje por terminal, como muestra la siguiente imagen:

```
[ INFO] [1537104615.469035132]: 329-637-1-1-0
[ INFO] [1537104615.469137404]: 638-652-1-2-0
[ INFO] [1537104615.470290107]: 947-685-1-3-0
[ INFO] [1537104615.570319153]: 319-314-2-1-0
[ INFO] [1537104615.670304157]: 699-285-2-2-0
[ INFO] [1537104615.770320994]: 992-305-2-3-0
```

Ilustración 72. Mensajes que se generan tras enviar información a un topic, en ROS.

4.3.4 Generación rutas solución.

Una vez recibidas las celdas donde aparecen minas, haciendo uso de la metodología backtracking generaba las rutas solución. Por cada celda con minas se generaba una ruta solución, desde el origen hasta el destino (celda segura adyacente a la celda con minas). Partiendo de la representación del terreno con las siguientes minas en la siguiente matriz:

X		X
		X

Ilustración 73. Representación de un terreno de celdas con minas en alguna de ellas.

Donde las X representan celdas con minas, se obtienen las siguientes rutas solución:

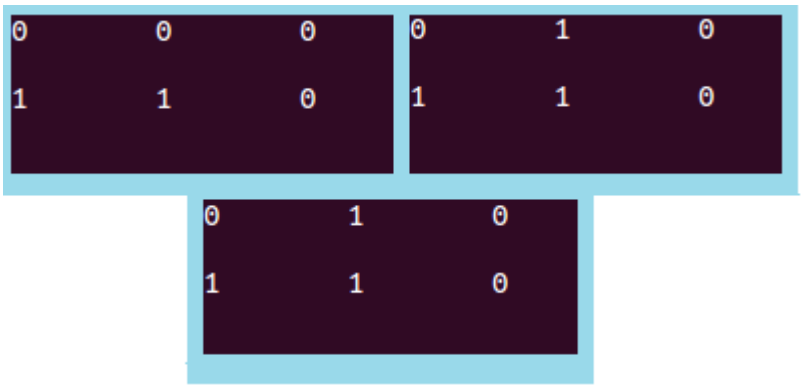


Ilustración 74. Rutas solución.

Estas rutas se envían de nuevo a un nuevo topic para proceder a la desactivación de las minas.

Quedaba validado así el algoritmo propuesto, por lo que finalizaría el estudio de más propuestas viables.

5. LINEAS FUTURAS DE INVESTIGACION Y CONCLUSIONES

El estado actual del Trabajo de Fin de Grado no está implementado con el resto de componentes que conforman la competición. Si bien se ha dejado de una forma muy genérica para que resulte sencilla su implementación, quedaría probar y verificar la validez del algoritmo.

Otra mejora posible al algoritmo seria poder determinar automáticamente, sin tener que calibrar previamente, los valores de HSV para hacer el separado de color de la imagen entrante. Con esto, podríamos utilizar el algoritmo en cualquier tipo de terreno, ya que este haría un barrido de todos los valores de HSV hasta que detectase todas las celdas del terreno, sin saltarse ninguna.

Una última mejora podría ser el seguimiento en vivo, por parte del UAV, del terreno a clasificar. Esto quiere decir que no haría falta sacar una foto previa del terreno, para después clasificarla. Sería el propio dron el que transmitiese constantemente una imagen del terreno, y este estaría clasificando constantemente las celdas. Con esto, contribuimos a poder desarrollar una aplicación que sea automática, pues ahora tenemos que centrar el UAV para poder tomar una imagen correcta del terreno. De esta nueva forma, sería el dron el que seleccionase el mejor ángulo para tomar la foto.

En conclusión, este Trabajo de Fin de grado se ha centrado en la detección y clasificación de un terreno con celdas a partir de una imagen fija, para luego elaborar unas rutas seguras para la desactivación de las minas contenidas en ellas

6. ANEXOS

6.1 Código para la selección de valores de HSV de usuario anónimo.

```
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <cv.h>
#include <highgui.h>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{

/*
VideoCapture cap(0); //capture the video from web cam

if ( !cap.isOpened() ) // if not success, exit program
{
cout << "Cannot open the web cam" << endl;
return -1;
}*/

namedWindow("Control", CV_WINDOW_AUTOSIZE); //create a window
called "Control"

int iLowH = 0;
int iHighH = 179;

int iLowS = 0;
int iHighS = 255;

int iLowV = 0;
int iHighV = 255;

//Create trackbars in "Control" window
cvCreateTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
cvCreateTrackbar("HighH", "Control", &iHighH, 179);

cvCreateTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 -
255)
cvCreateTrackbar("HighS", "Control", &iHighS, 255);

cvCreateTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 255)
cvCreateTrackbar("HighV", "Control", &iHighV, 255);
```

```
while (true)
{

Mat imgOriginal = imread("prueba1.jpg", CV_LOAD_IMAGE_COLOR);
Mat imgHSV;

cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert the
captured frame from BGR to HSV

Mat imgThresholded;

medianBlur(imgHSV, imgHSV, 11);

inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS,
iHighV), imgThresholded); //Threshold the image

//morphological closing (fill small holes in the foreground)
dilate(imgThresholded, imgThresholded,
getStructuringElement(MORPH_RECT, Size(20, 20)));
erode(imgThresholded, imgThresholded,
getStructuringElement(MORPH_RECT, Size(20, 20)));

//morphological opening (remove small objects from the foreground)
erode(imgThresholded, imgThresholded,
getStructuringElement(MORPH_RECT, Size(20, 20)));
dilate(imgThresholded, imgThresholded,
getStructuringElement(MORPH_RECT, Size(20, 20)));

imshow("Thresholded Image", imgThresholded); //show the thresholded
image
imshow("Original", imgOriginal); //show the original image

if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If
'esc' key is pressed, break loop
{
cout << "esc key is pressed by user" << endl;
}
}

return 0;
}
```

7. BIBLIOGRAFIA

7.1 URL's de Páginas WEB

http://www.pensamientoscomputables.com/entrada/por-que-modelo-color-rgb.html **(Agosto 2018)**
https://es.wikipedia.org/wiki/Fotografía_en_blanco_y_negro **(Agosto 2018)**
https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf **(Agosto 2018)**
https://www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny **(Agosto 2018)**
http://oefa.blogspot.com/2009/04/deteccion-de-bordes-algoritmo-de-canny.html**(Agosto 2018)**
https://joanflo.wordpress.com/2014/10/21/backtracking-para-resolver-puzzles-de-puntos-y-su-coste-computacional/ **(Agosto 2018)**
https://www.publico.es/internacional/paises-siguen-matando-minas-antipersona.html **(Agosto 2018)**
https://www.yamahamotorsports.com/motorsports/pages/precision-agriculture **(Agosto 2018)**
https://gust.com/companies/b-droid **(Agosto 2018)**
http://www.rpaslife.es/2016/01/23/drones-herramientas-de-busqueda-y-rescate/ **(Agosto 2018)**
http://flightechspanish.weebly.com/vigilancia.html **(Agosto 2018)**
https://visartblog.wordpress.com/2013/04/24/uav-militar-usa-vision-artificial/ **(Agosto 2018)**
http://rm-forwarding.com/2018/01/12/uav-carga-presentado-boeing/ **(Agosto 2018)**
https://www.popsci.com/china-sharp-sword-lijian-stealth-drone **(Agosto 2018)**
http://www.xdrones.es/los-5-drones-militares-que-china-esta-desarrollando-en-las-sombras/ **(Agosto 2018)**

7.2 Artículos Científicos

ErliangYao, HexinZhang, HuiXu, HaitaoSong, GuoliangZhangb. **Robust RGB-D visual odometry based on edges and points. (Acceso Junio 2018).**
https://www.sciencedirect.com/science/article/pii/S0921889018300770

Leonardo De-Maeztu, Unai Elordi, Marcos Nieto, Javier Barandiaran, Oihana Otaegui.**A temporally consistent grid-based visual odometry framework for multi-core architectures. (Acceso Junio 2018).**

https://link.springer.com/article/10.1007/s11554-014-0425-y

Han Wang, Wei Mou, Gerald Seet, Mao-Hai Li, M. W. S. Lau, Dan-Wei Wang. **Real-time Visual Odometry Estimation Based on Principal Direction Detection on Ceiling Vision. (Acceso Junio 2018).**

https://www.sciencedirect.com/science/article/pii/S0921889015303183

Zhao Jun, Liu Guo-ping**. A novel localization method for indoor mobile robot based on odometry and ceiling visual features. (Acceso Junio 2018).**

https://www.semanticscholar.org/paper/A-novel-localization-method-for-indoor-mobile-robot-Jun-Guo-ping/2c29ae73ec5199402a63b73ff2d8310bd620670e